Deep Neural Networks II

Sen Wang

UDRC Co-I – WP3.1 and WP3.2
Assistant Professor in Robotics and Autonomous Systems
Institute of Signals, Sensors and Systems
Heriot-Watt University

UDRC-EURASIP Summer School
26th June 2019 - Edinburgh

Slides adapted from Andrej Karpathy, Kaiming He

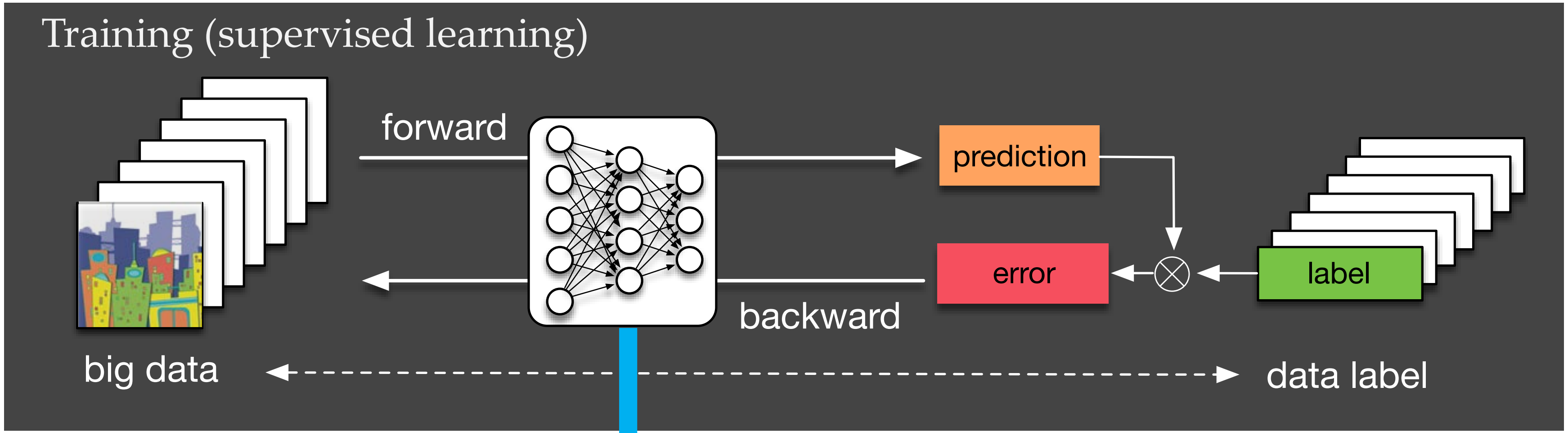
Outline

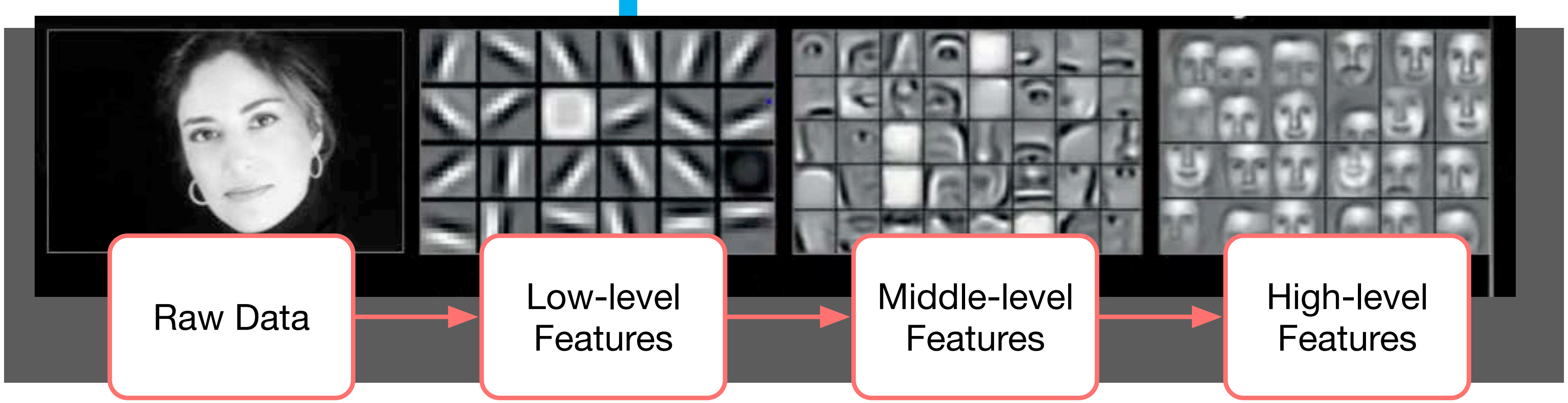
Learning features for machines to solve problems

- Convolutional Neural Networks (CNNs)
- Deep Learning Architectures (focus on CNNs) - **learning features**
- Some Deep Learning Applications - **problems**
  - Object detection (image, radar, sonar)
  - Semantic segmentation
  - Visual odometry
  - 3D reconstruction
  - Semantic mapping
  - Robot navigation
  - Manipulation and grasping
  - …………

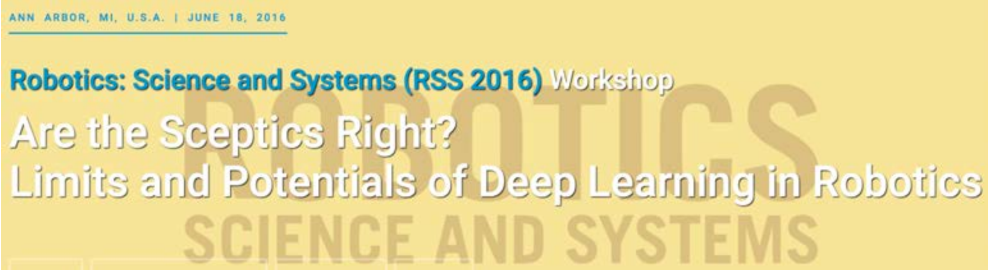
Deep Learning

Deep Learning: a learning technique combining layers of neural networks to **automatically identify features** that are relevant to the problem to solve





Deep Learning in Robotics



ANN ARBOR, MI, U.S.A. | JUNE 18, 2016

Robotics: Science and Systems (RSS 2016) Workshop

Are the Sceptics Right?
Limits and Potentials of Deep Learning in Robotics

BOSTON, MA, U.S.A. | JULY, 2017

Robotics: Science and Systems (RSS 2017) Workshop

New Frontiers for Deep Learning in Robotics

HONOLULU, HI, U.S.A. | 21 JULY, 2017

Computer Vision and Pattern Recognition (CVPR 2017) Workshop

Deep Learning for Robotic Vision

Call for Papers: ***The International Journal of Robotics Research*** **(IJRR)**

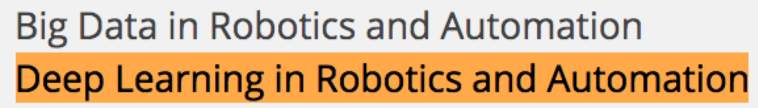
Special Issue: *Limits and Potentials of Deep Learning in Robotics*

IJRR 2016

International Journal of Computer Vision

Special Issue on **Deep Learning for Robotic Vision**

IJCV 2018



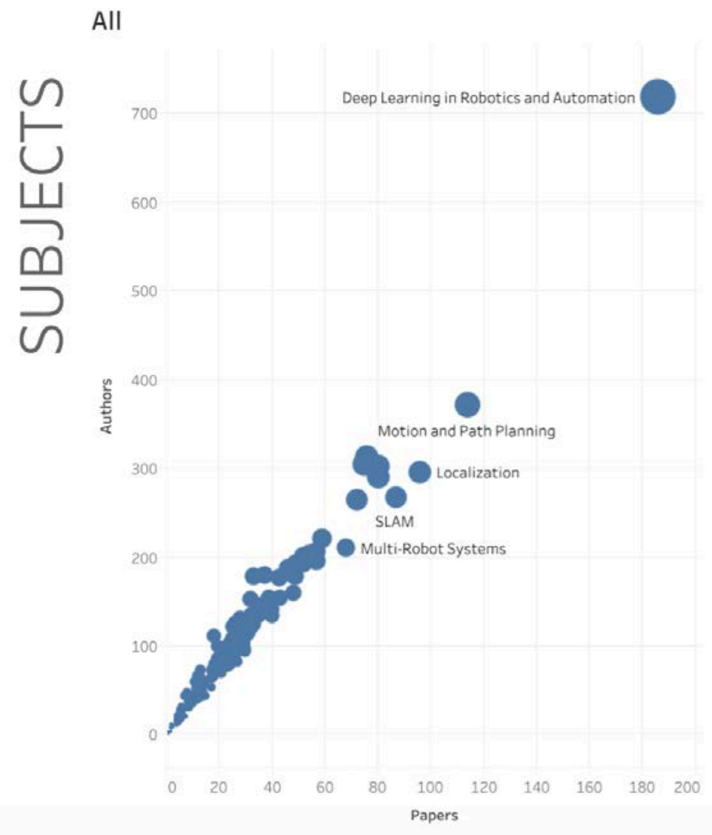
Big Data in Robotics and Automation

Deep Learning in Robotics and Automation

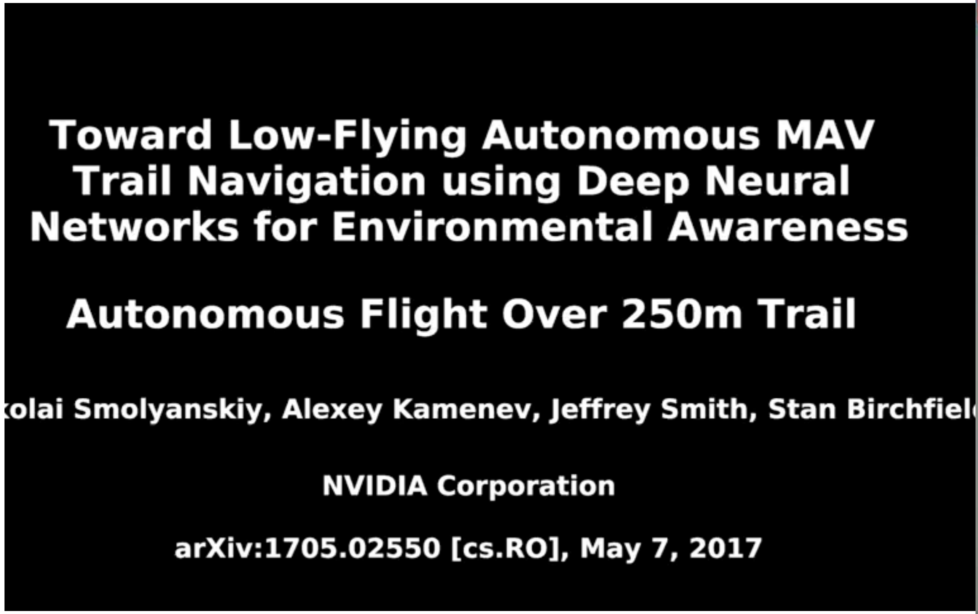
ICRA2018 ~2500 submissions:

the most popular keyword

ICRA 2019



Deep Learning in Robotics



Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness

Autonomous Flight Over 250m Trail

olai Smolyanskiy, Alexey Kamenev, Jeffrey Smith, Stan Birchfiel

NVIDIA Corporation

arXiv:1705.02550 [cs.RO], May 7, 2017

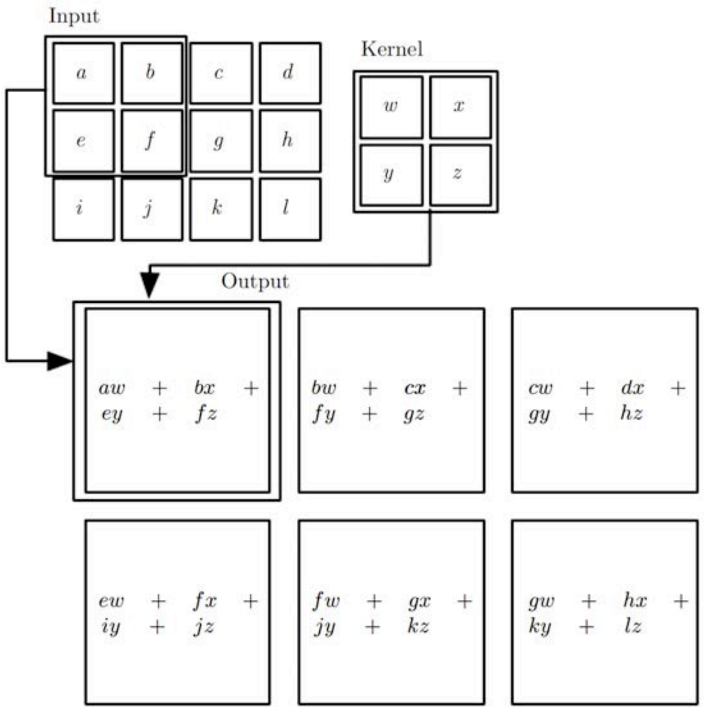
Convolutional Neural Networks (CNNs)

From MLPs to CNNs

- Feed-forward Neural Networks or Multi-Layer Perceptrons (MLPs)
    - many multiplications

- CNNs are similar to Feed-forward Neural Networks
    - convolution instead of general matrix multiplication

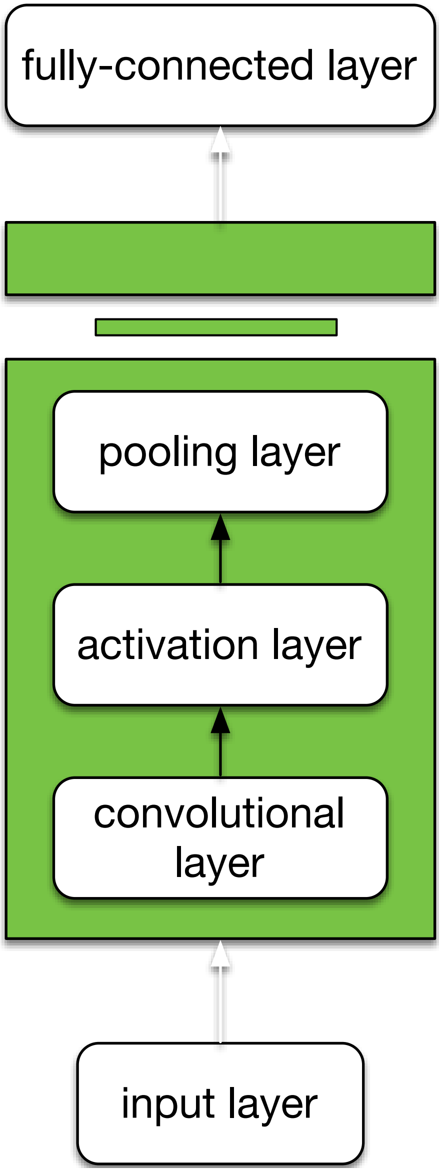
$$
\begin{aligned}
S(i,j) &= (I * K)(i,j) \\
&= \sum_m \sum_n I(i+m, j+n)K(m,n)
\end{aligned}
$$



CNNs

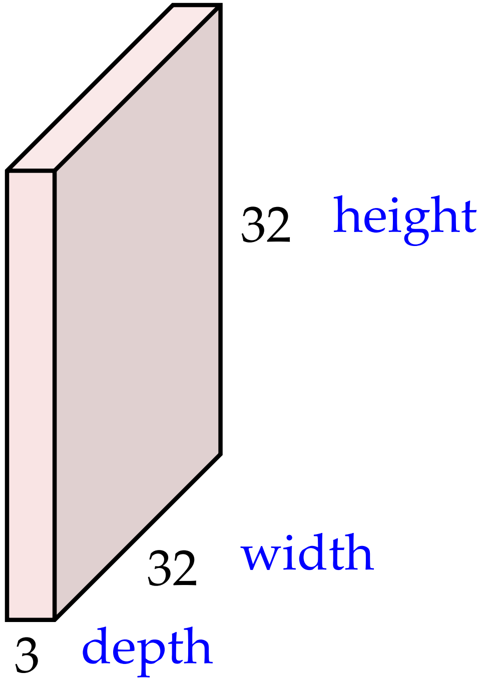
- 3 Main Types of Layers:
  - ○ **convolutional layer**
  - ○ **activation layer**
  - ○ **pooling layer**

- repeat many times

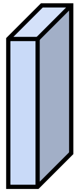


CNNs: Convolution Layer

32x32x3 image



5x5x3 filter

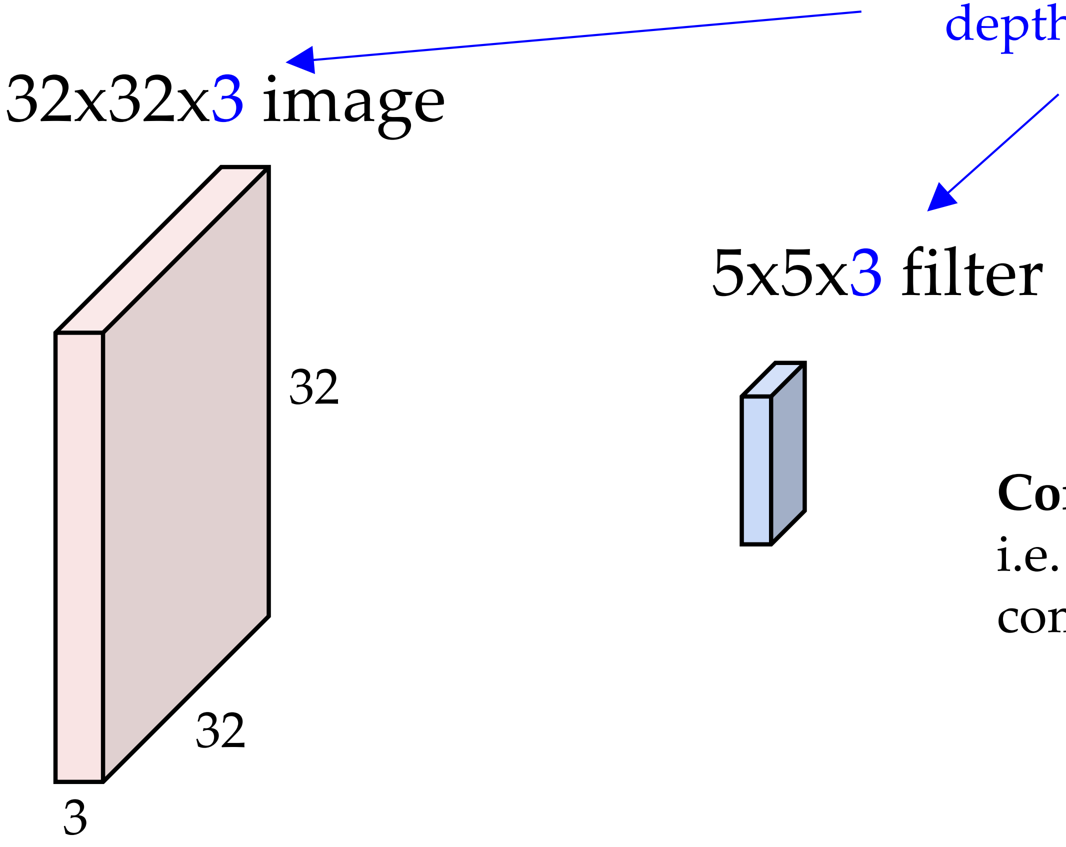


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Slides courtesy of Andrej Karpathy

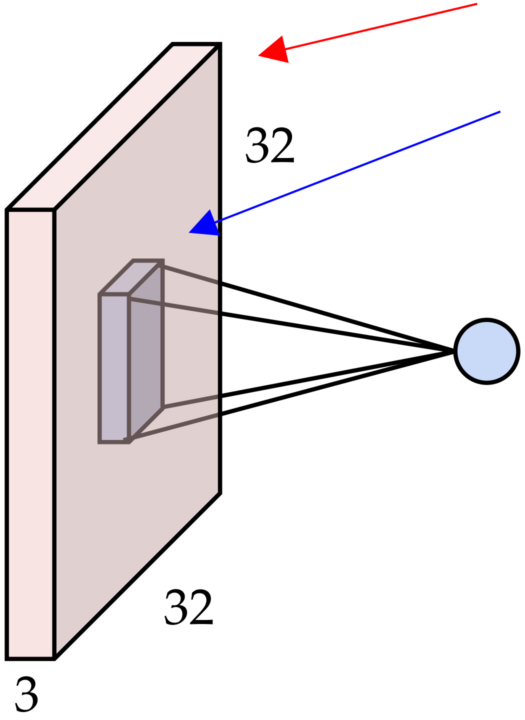
CNNs: Convolution Layer

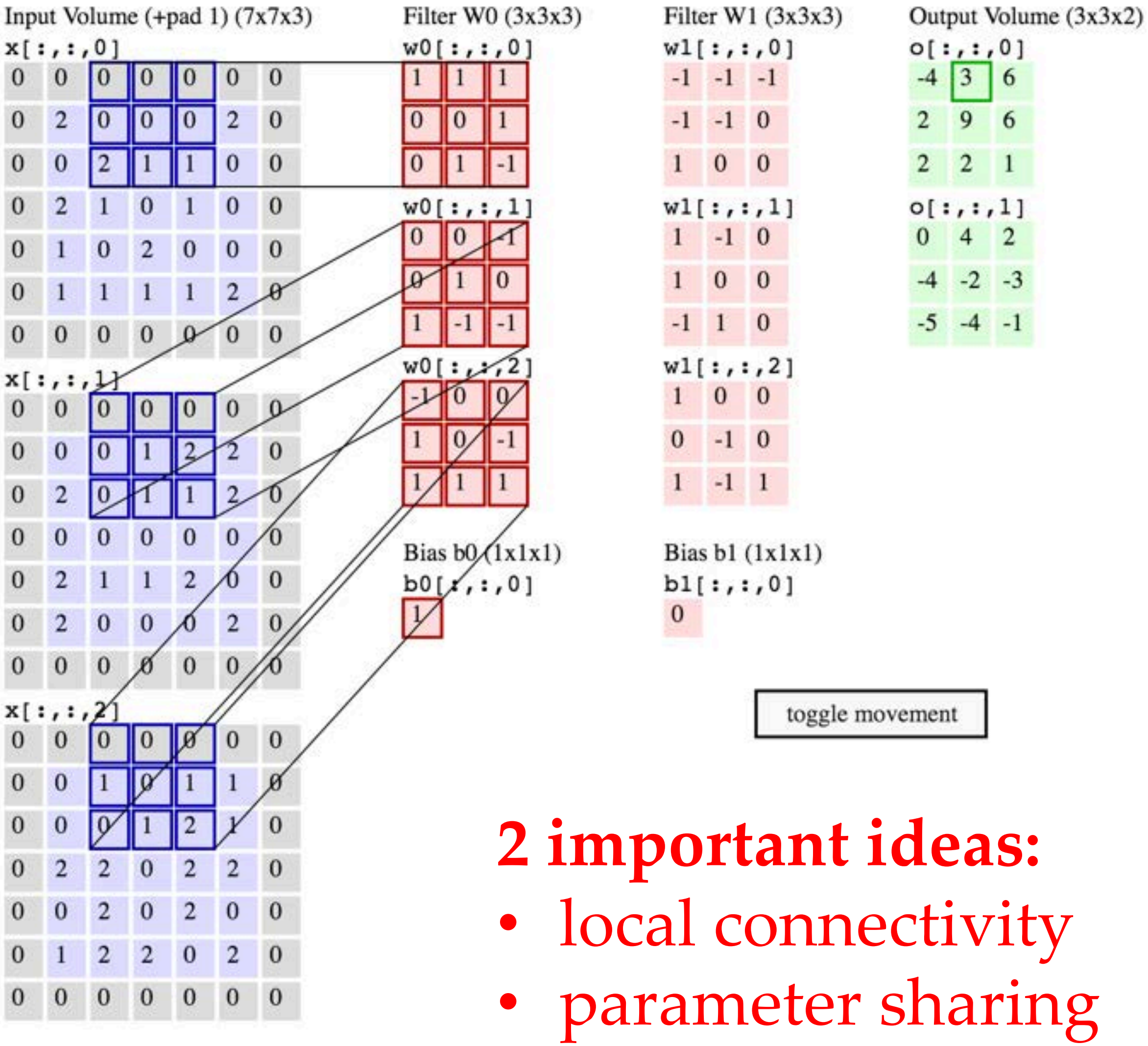
Filters always extend the full depth of the input volume



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

CNNs: Convolution Layer





x[:,:,0]

0	0	0	0	0	0	0
0	2	0	0	0	2	0
0	0	2	1	1	0	0
0	2	1	0	1	0	0
0	1	0	2	0	0	0
0	1	1	1	1	2	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	0	0	1	2	2	0
0	2	0	1	1	2	0
0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	2	0	0	0	2	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	1	0	1	1	0
0	0	0	1	2	1	0
0	2	2	0	2	2	0
0	0	2	0	2	0	0
0	1	2	2	0	2	0
0	0	0	0	0	0	0

w0[:,:,0]

1	1	1
0	0	1
0	1	-1

w0[:,:,1]

0	0	-1
0	1	0
1	-1	-1

w0[:,:,2]

-1	0	0
1	0	-1
1	1	1

b0[:,:,0]

1

w1[:,:,0]

-1	-1	-1
-1	-1	0
1	0	0

w1[:,:,1]

1	-1	0
1	0	0
-1	1	0

w1[:,:,2]

1	0	0
0	-1	0
1	-1	1

b1[:,:,0]

0

o[:,:,0]

-4	3	6
2	9	6
2	2	1

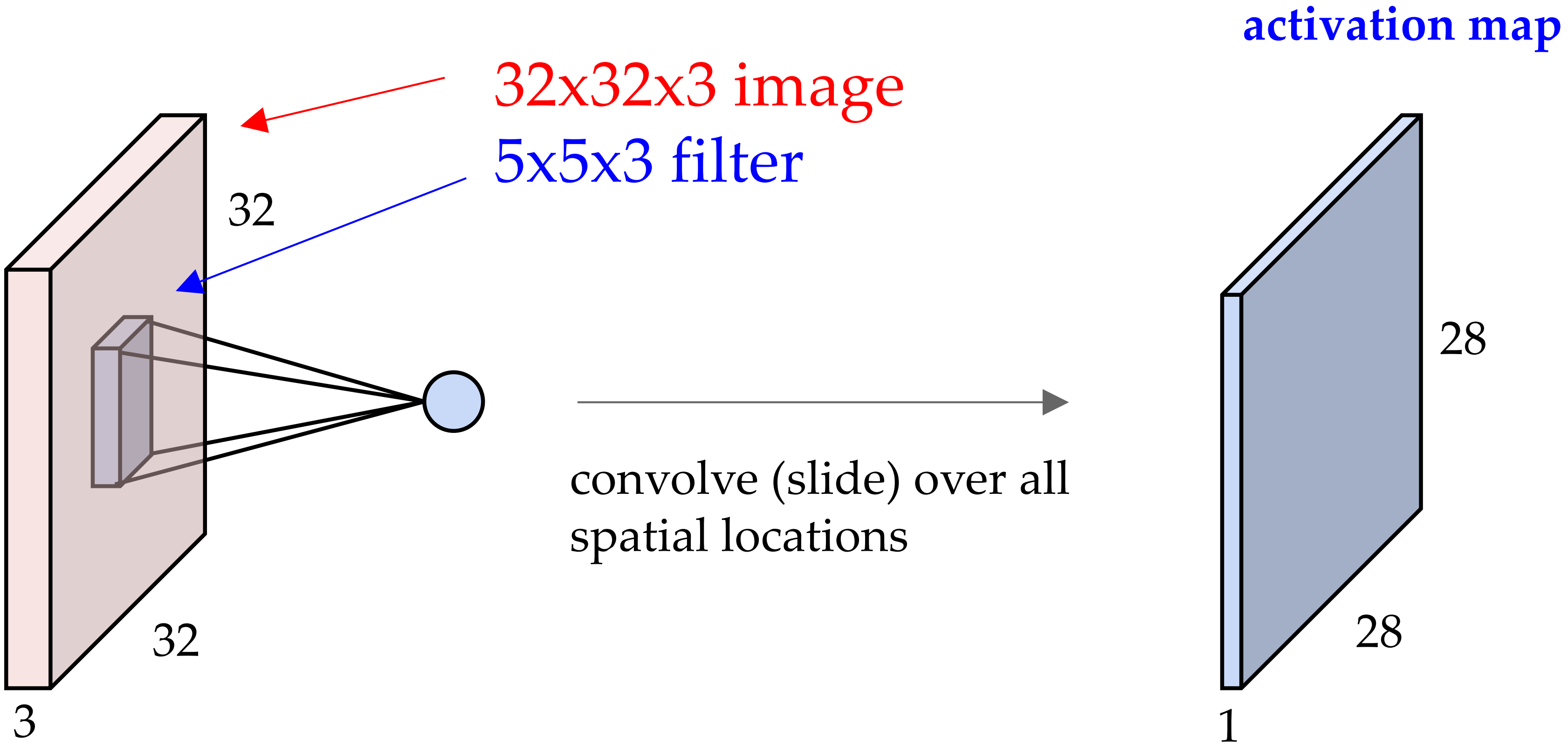
o[:,:,1]

0	4	2
-4	-2	-3
-5	-4	-1

2 important ideas:

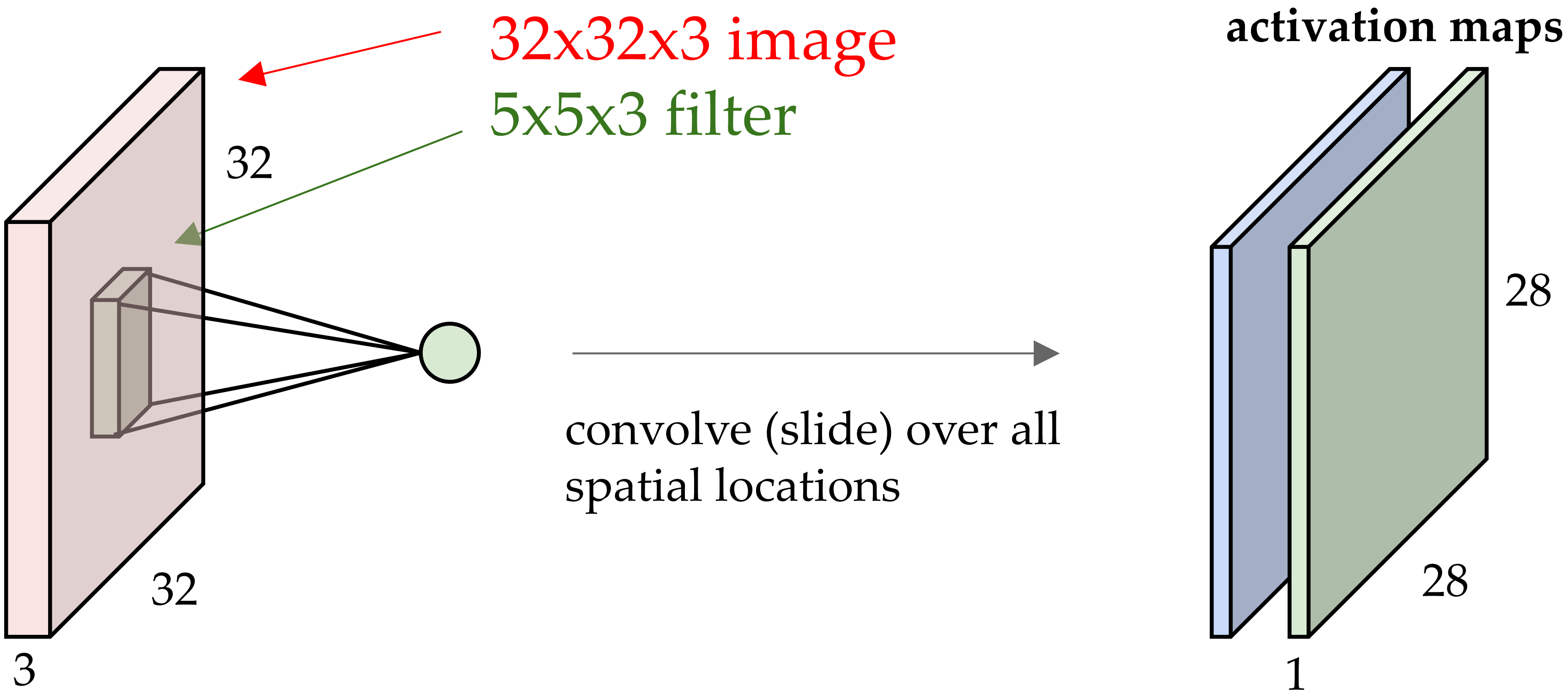
- local connectivity
- parameter sharing

CNNs: Convolution Layer



CNNs: Convolution Layer

consider a second, green filter



32x32x3 image

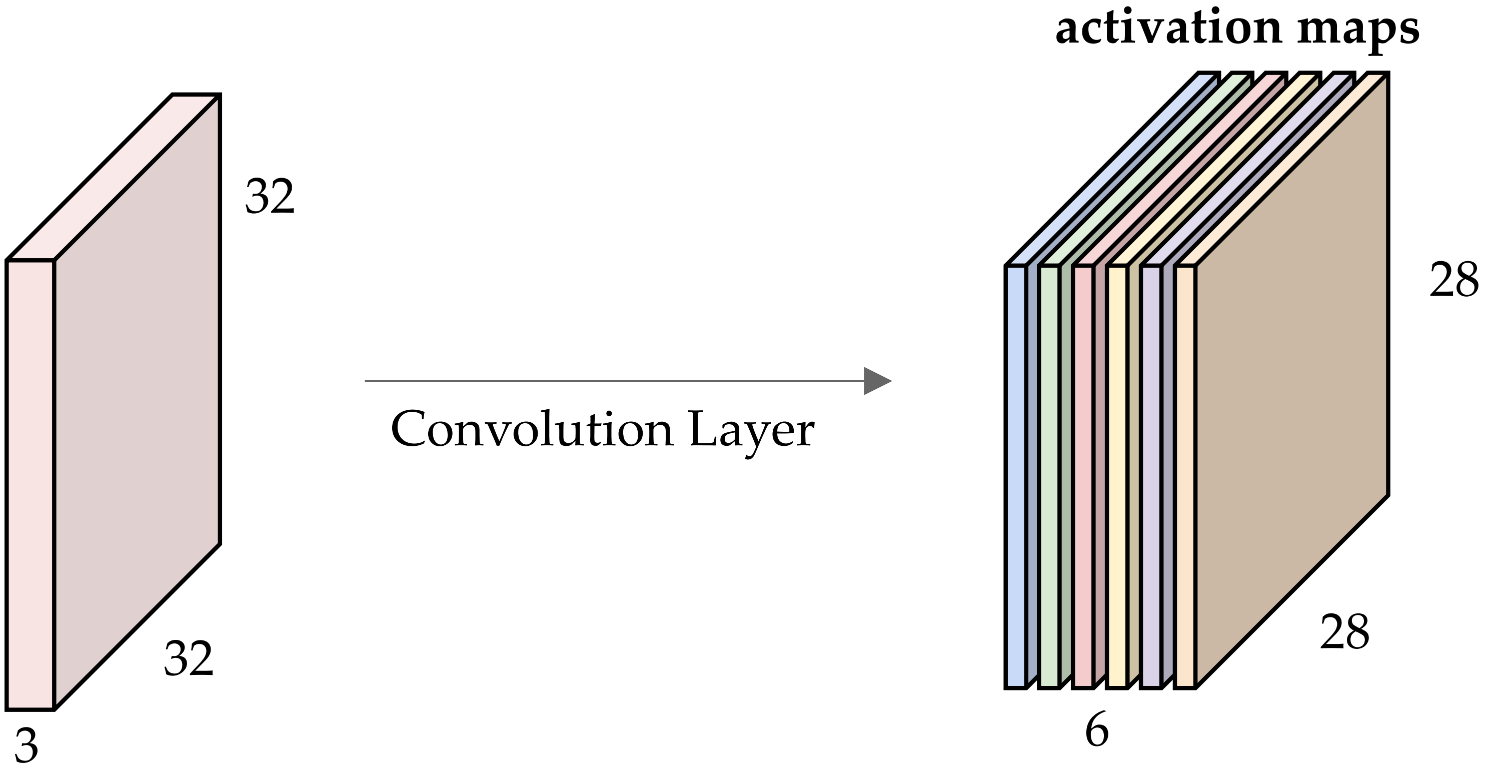
5x5x3 filter

activation maps

convolve (slide) over all spatial locations

CNNs: Convolution Layer

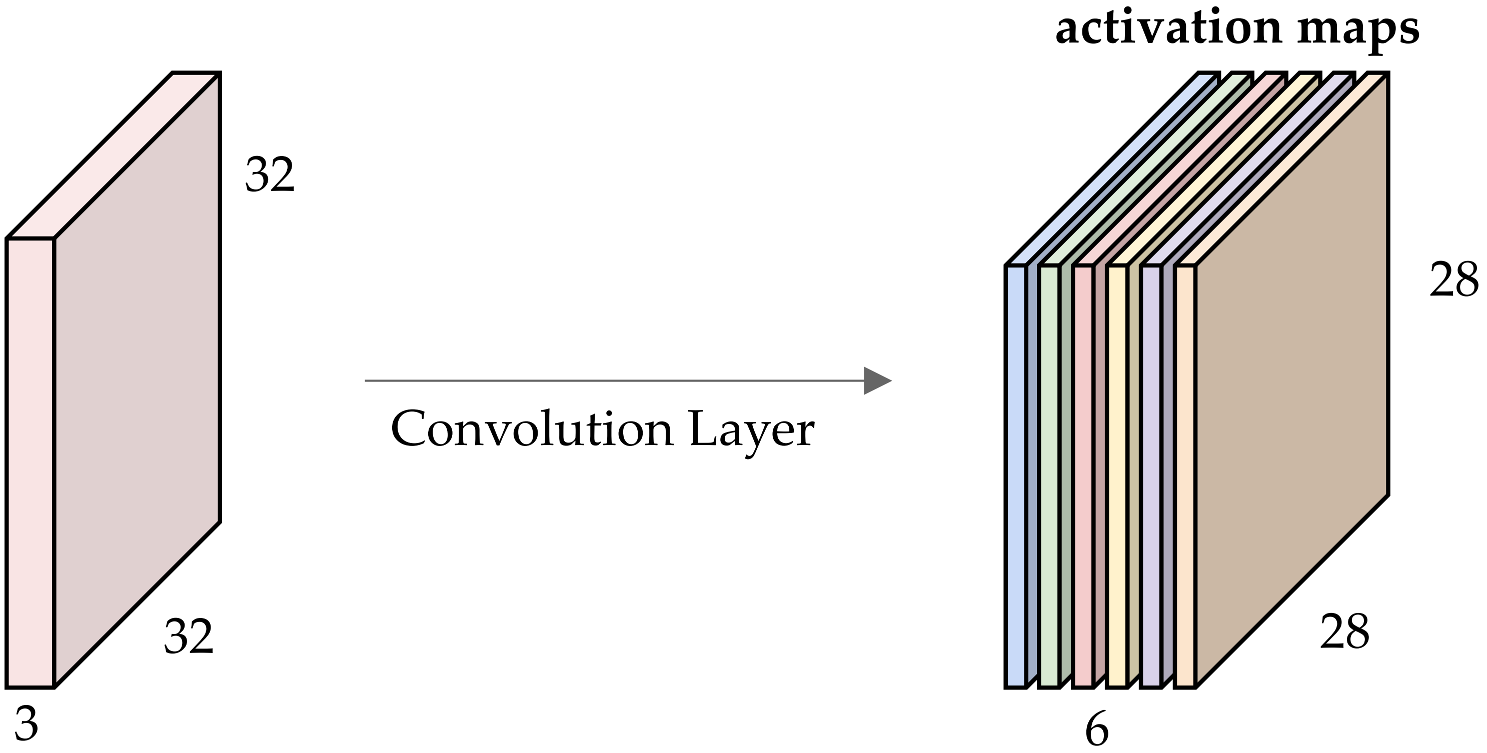
For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6

CNNs: Convolution Layer

For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:



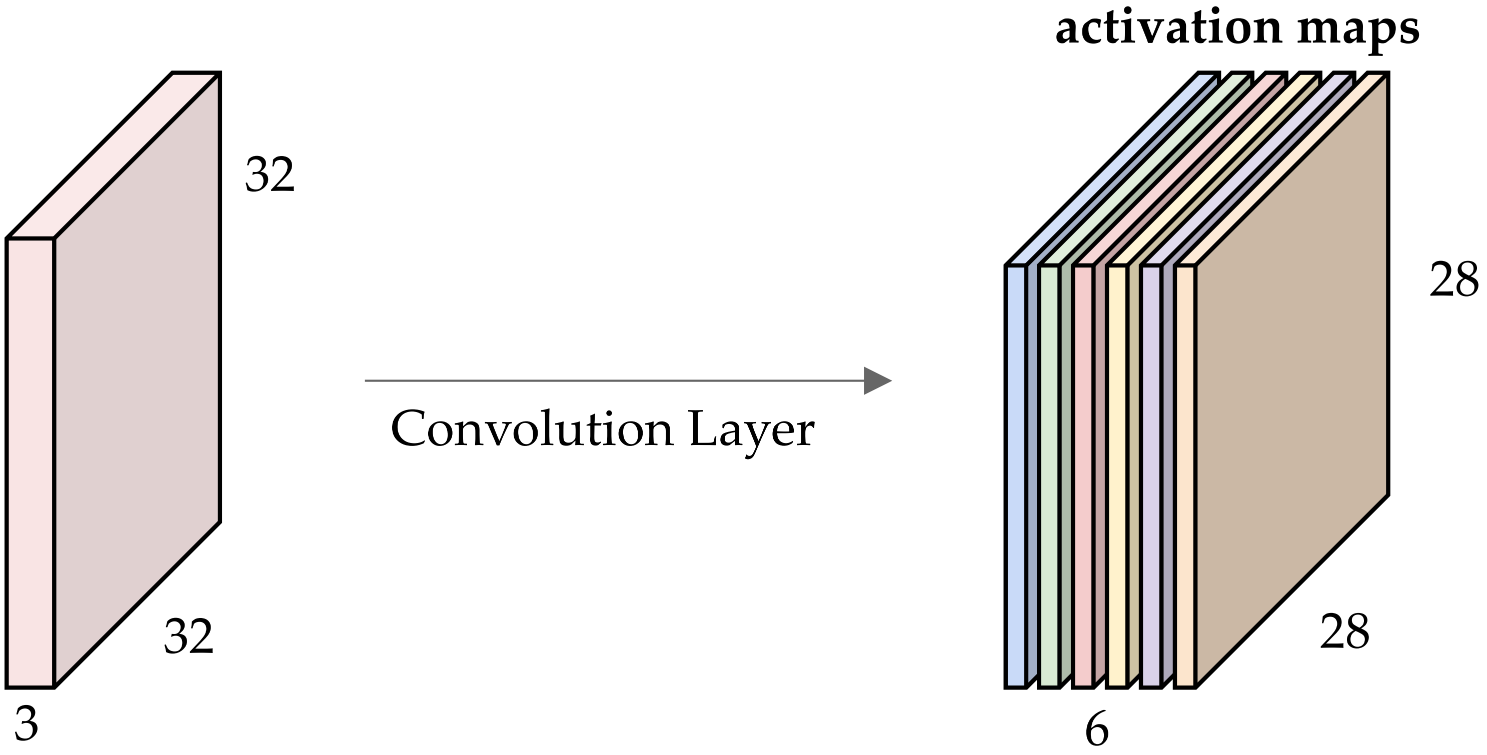
We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters would this be if we used a fully connected layer instead?

courtesy of Andrej Karpathy

CNNs: Convolution Layer

For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:



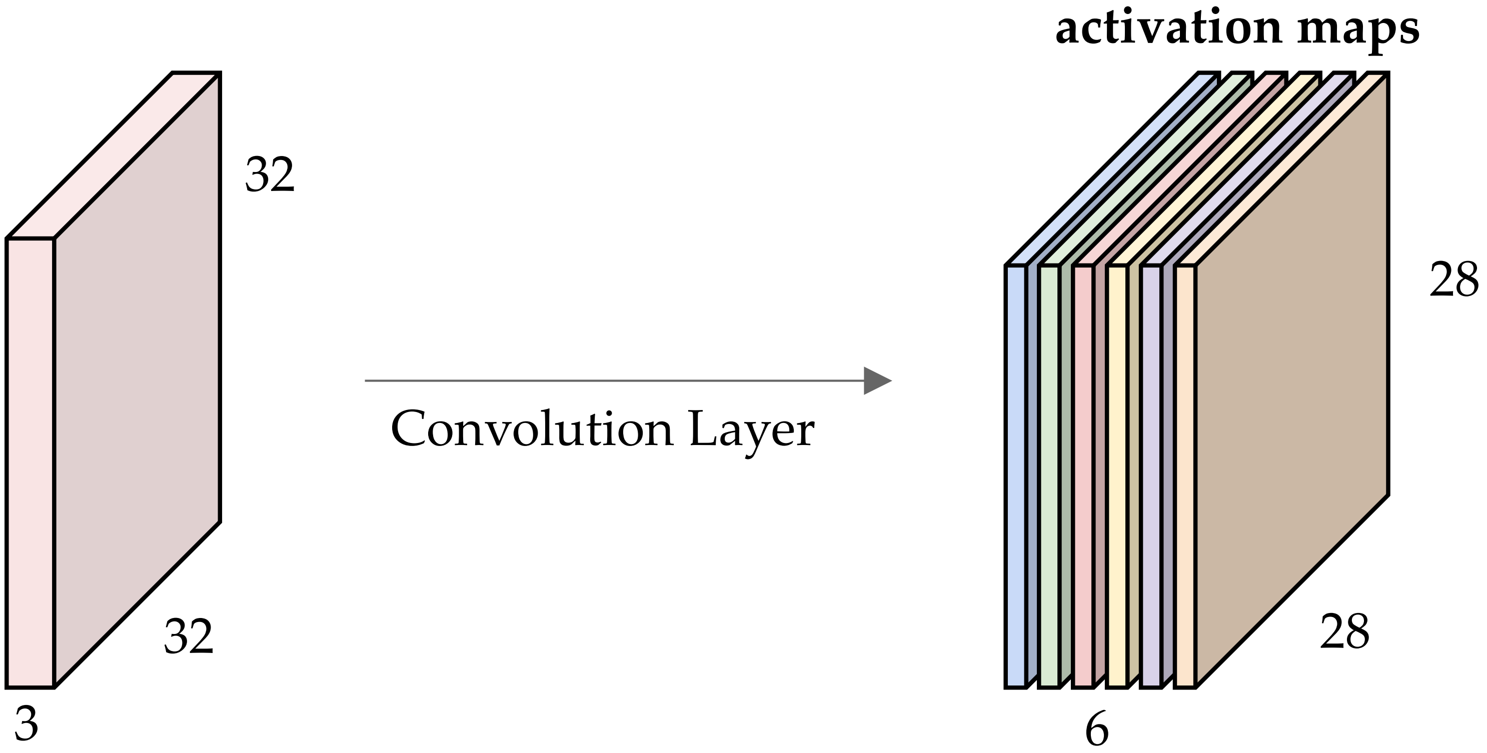
We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters would this be if we used a fully connected layer instead?

A: (32*32*3)*(28*28*6) = **14.5M parameters**, ~**14.5M multiplies**

CNNs: Convolution Layer

For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:

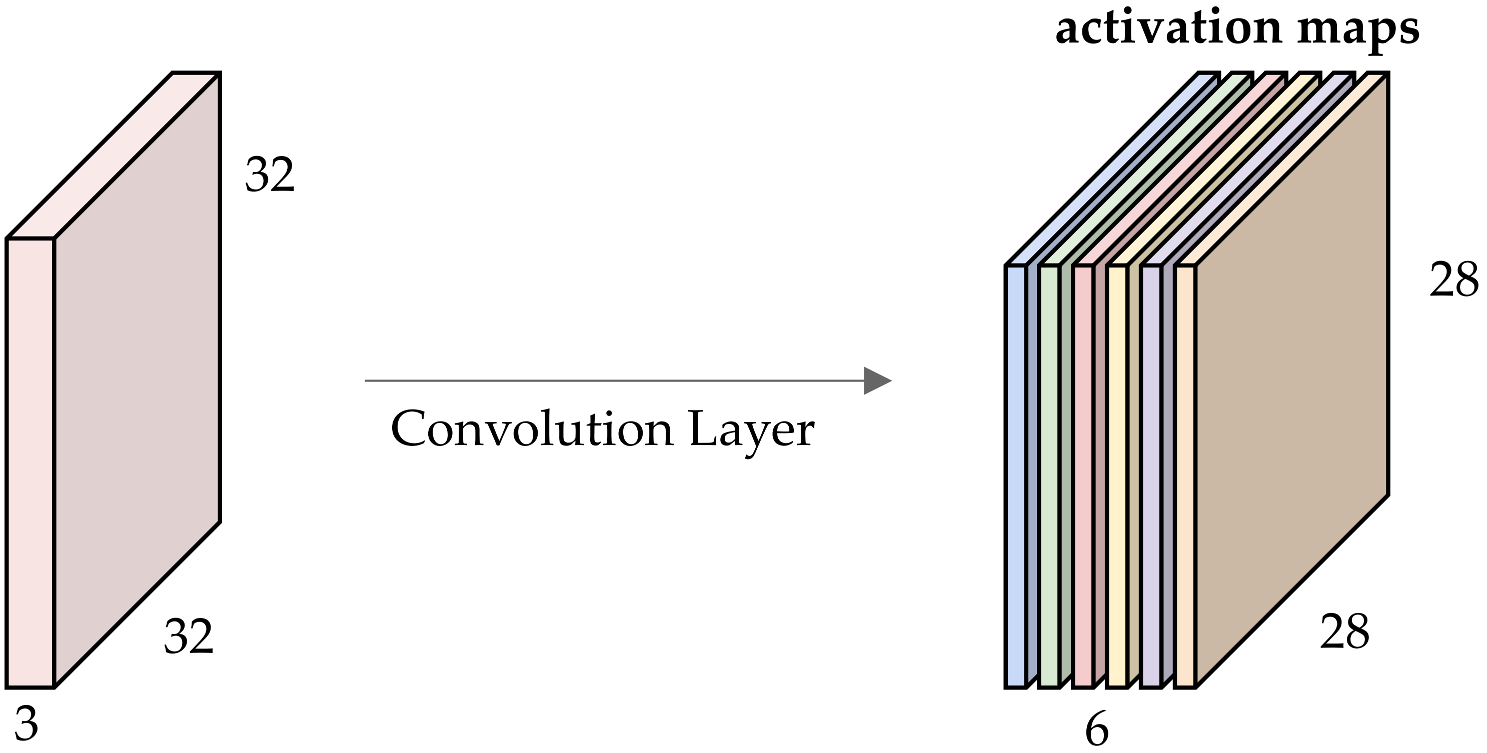


We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters are used instead?

CNNs: Convolution Layer

For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:



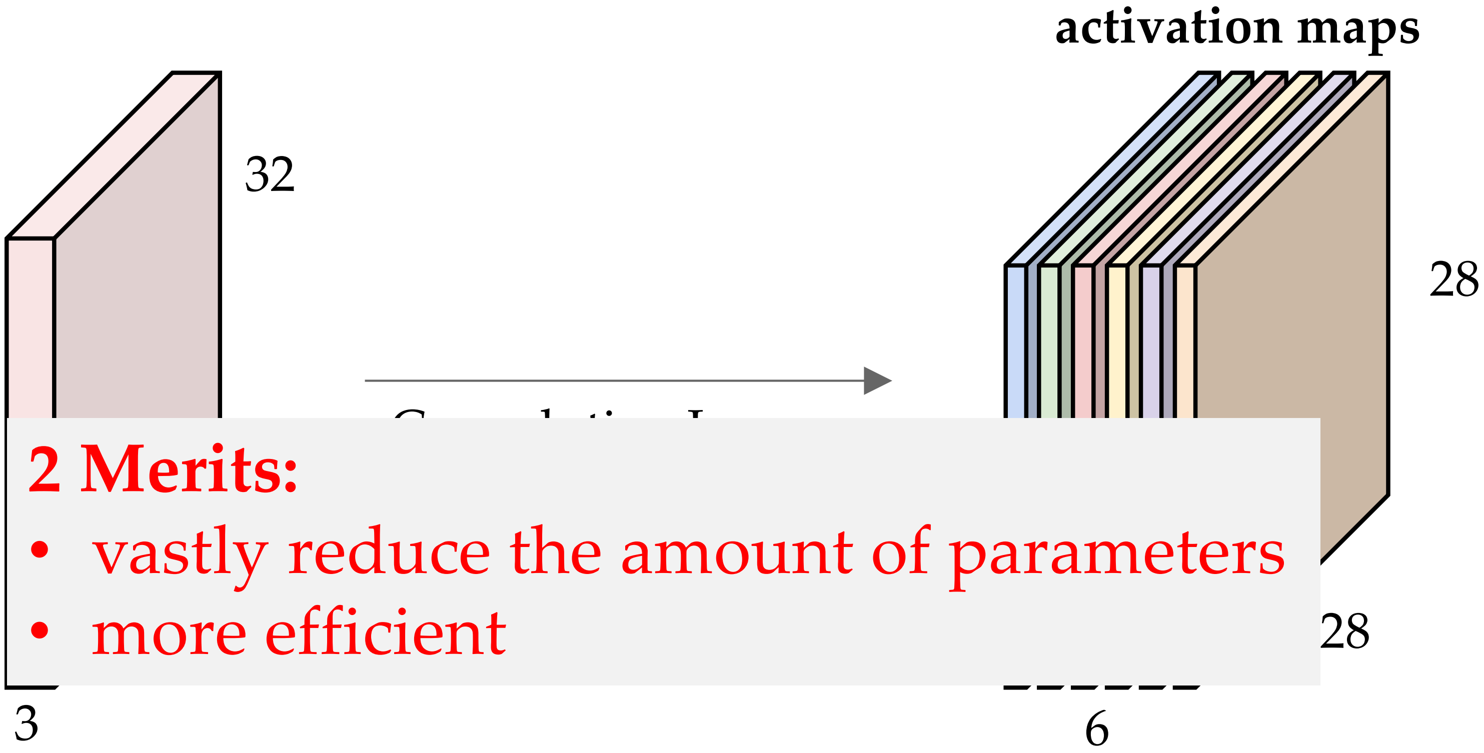
We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters are used instead? --- And how many multiplies?

A: (5*5*3)*6 = **450 parameters**

CNNs: Convolution Layer

For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:



2 Merits:

- vastly reduce the amount of parameters
- more efficient

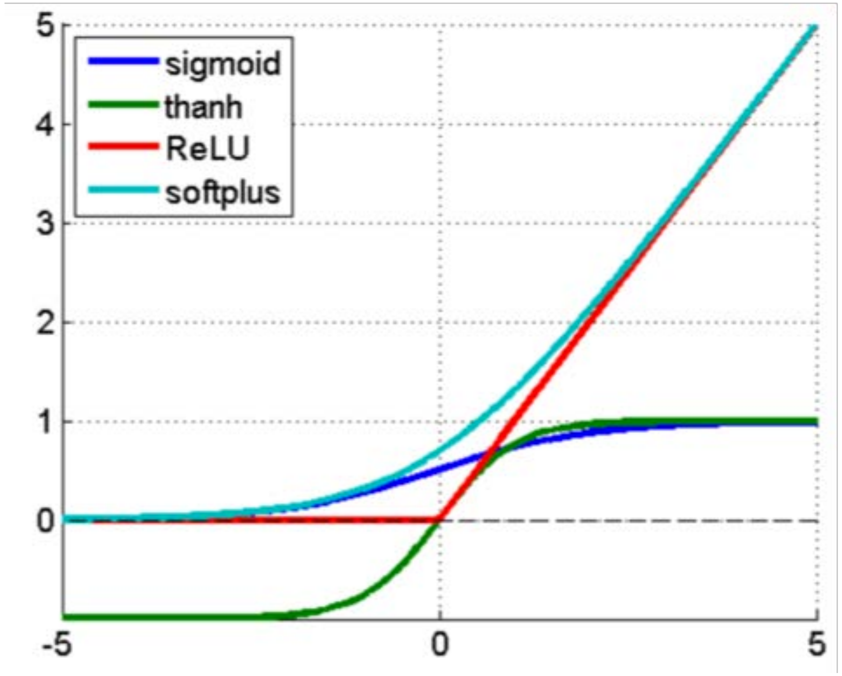
We processed [32x32x3] volume into [28x28x6] volume.

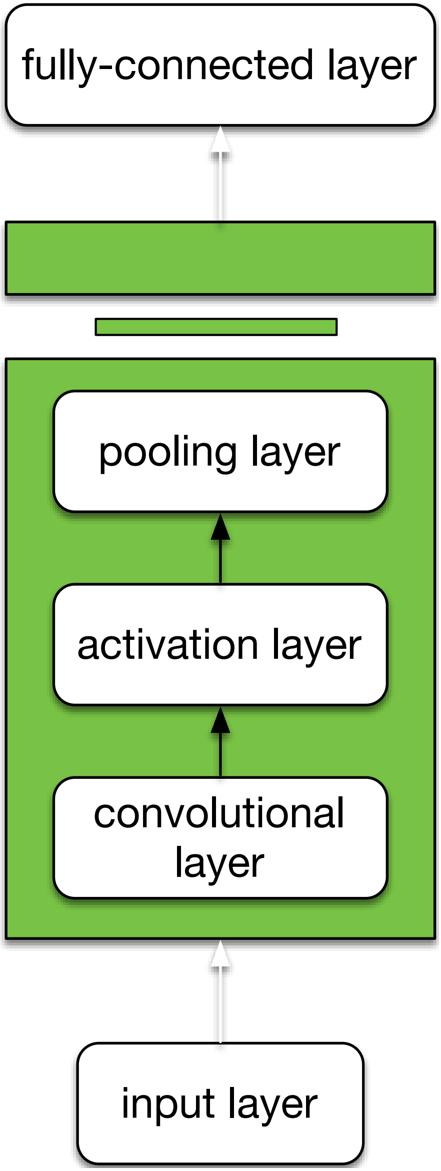
Q: how many parameters are used instead?

A: (5*5*3)*6 = **450 parameters**, (5*5*3)*(28*28*6) = **~350K multiplies**

CNNs: Activation Layer

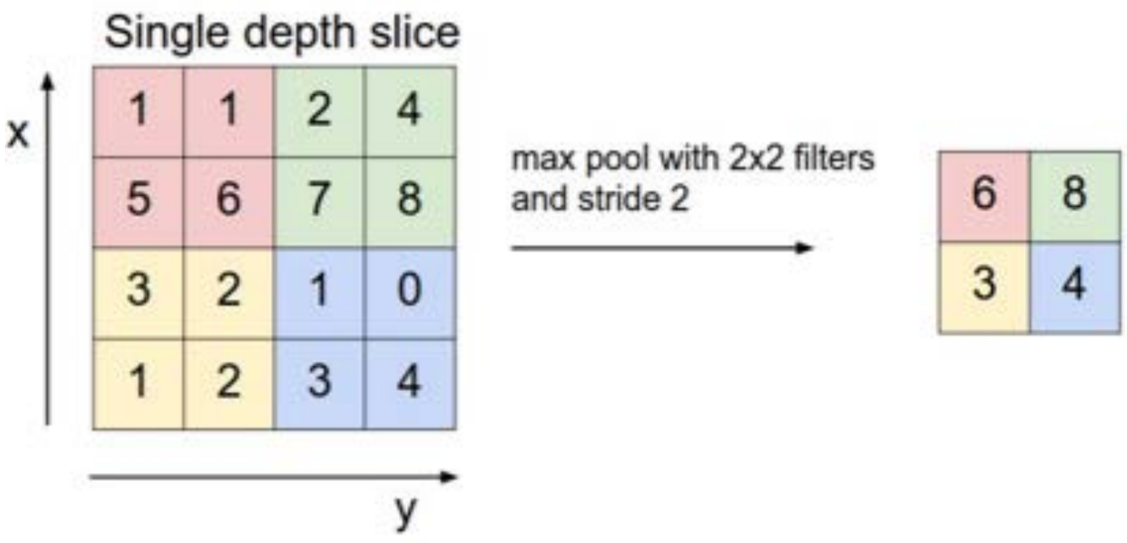
- 3 Main Types of Layers:
  - convolutional layer
  - **activation layer**
  - **pooling layer**





CNNs: Pooling Layer

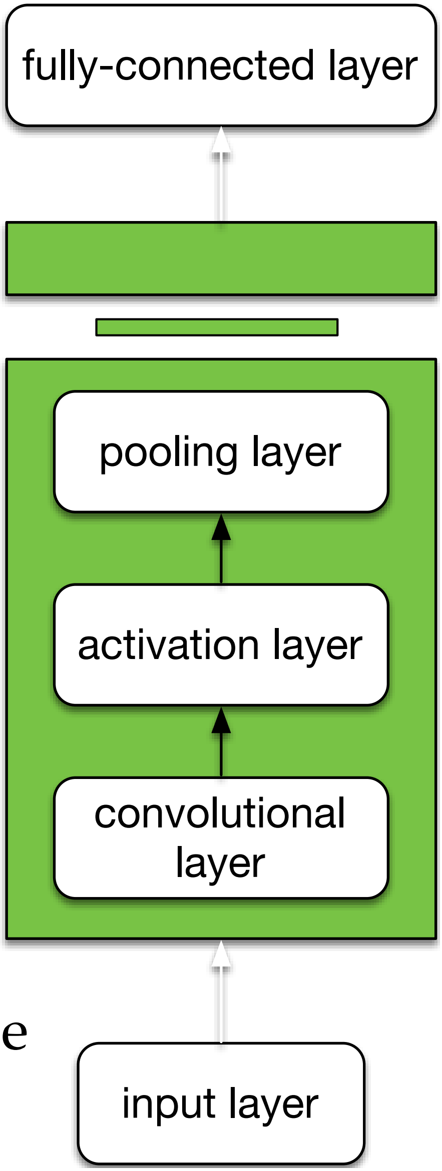
- 3 Main Types of Layers:
  - convolutional layer
  - activation layer
  - **pooling layer**



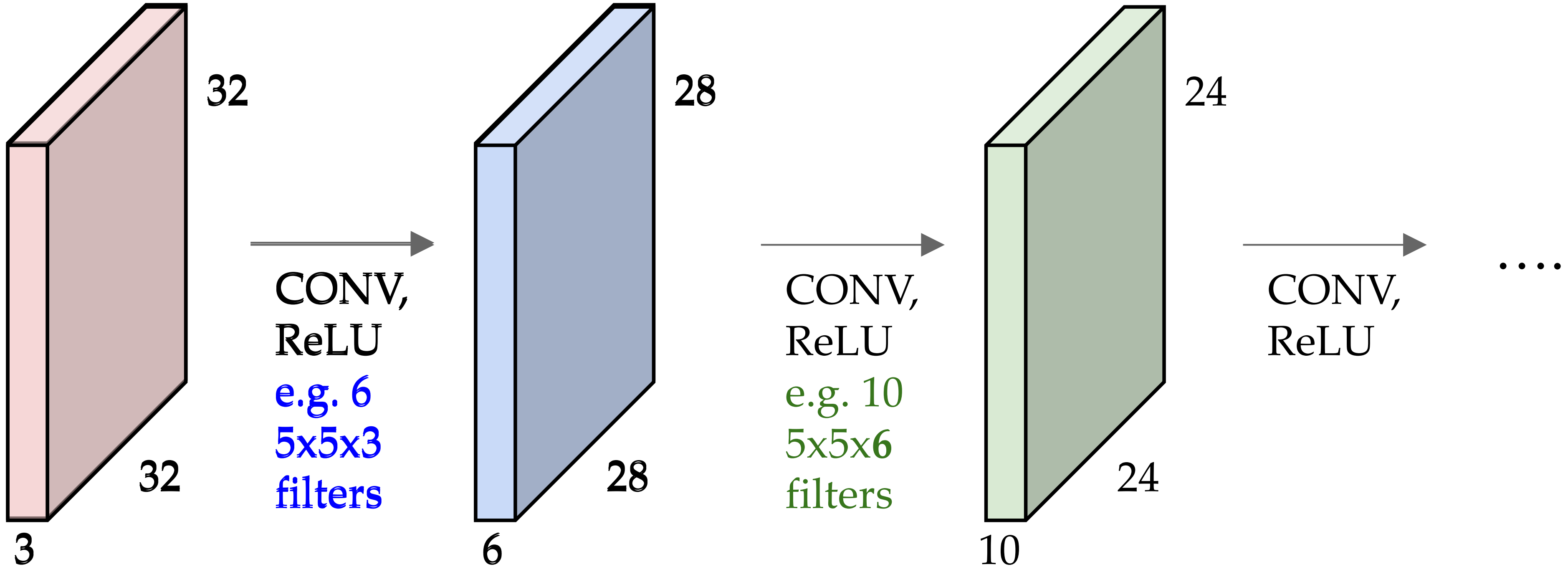
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

makes the representations smaller and more manageable

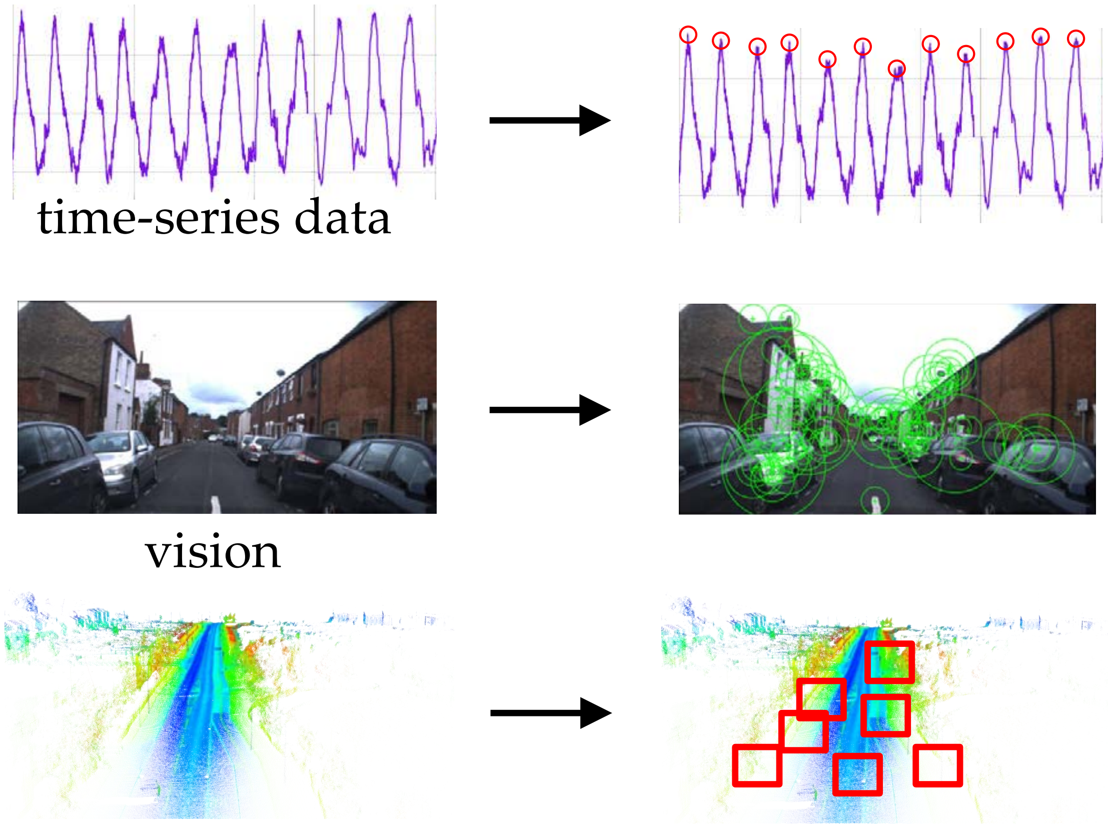


CNNs: A sequence of Convolutional Layers



Deep Learning Architectures

Hand-Crafted Features by Human



Pervasive Data

Feature Extraction (hand-crafted)

Inference

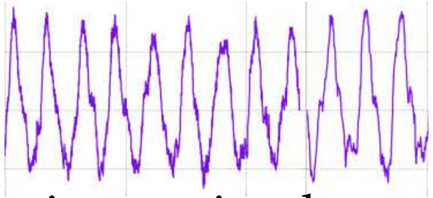
time-series data → → Activities, Context, …

vision → → Locations, Scene types, Semantics, …

point cloud → → Objects, Structure, …

Feature Engineering and Representation

Pervasive Data



time-series data

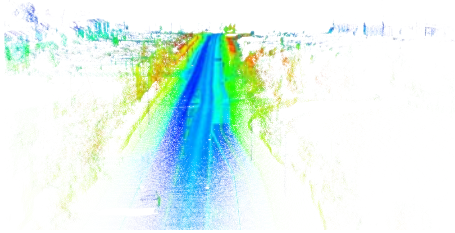
vision

$256^{3\text{x}800\text{x}600}$

Raw data

≈

Bad Representation

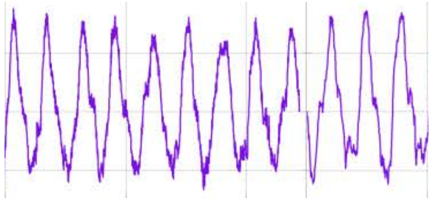


point cloud

$2^{?\text{x}?\text{x}?}$

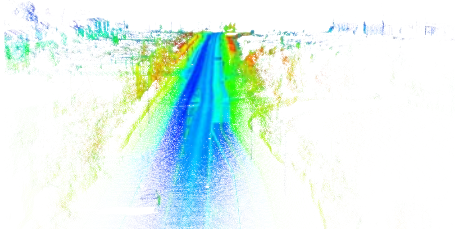
Deep Learning: Representation Learning

Pervasive Data



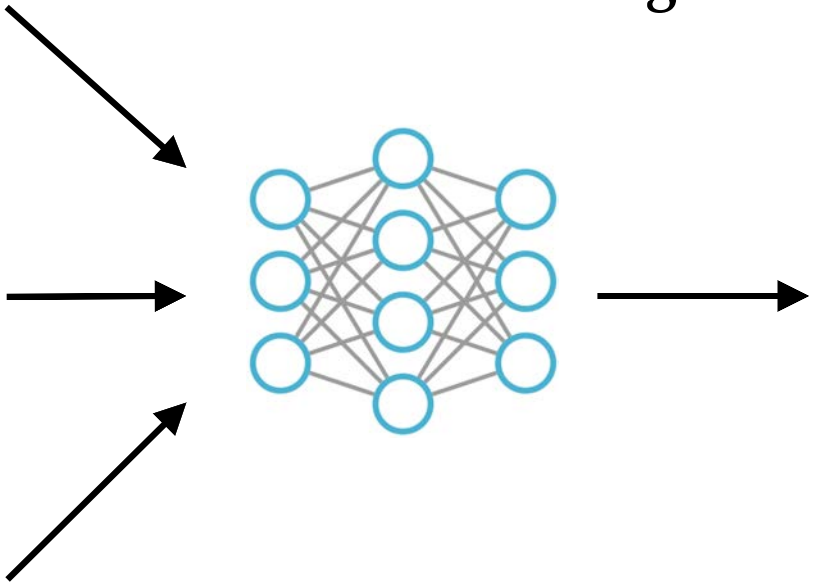
time-series data

vision



point cloud

End-to-End Learning



Inference

Activities,
Context, …
Locations,
Scene types, …
Structure,
Semantics, …

automatically learn effective feature representation to solve the problem

LeNet - 1998

- Convolution:
    - o **locally-connected**
    - o **spatially weight-sharing**
- weight-sharing is a key in DL
- Subsampling
- Fully-connected outputs

Foundation of modern ConvNets!

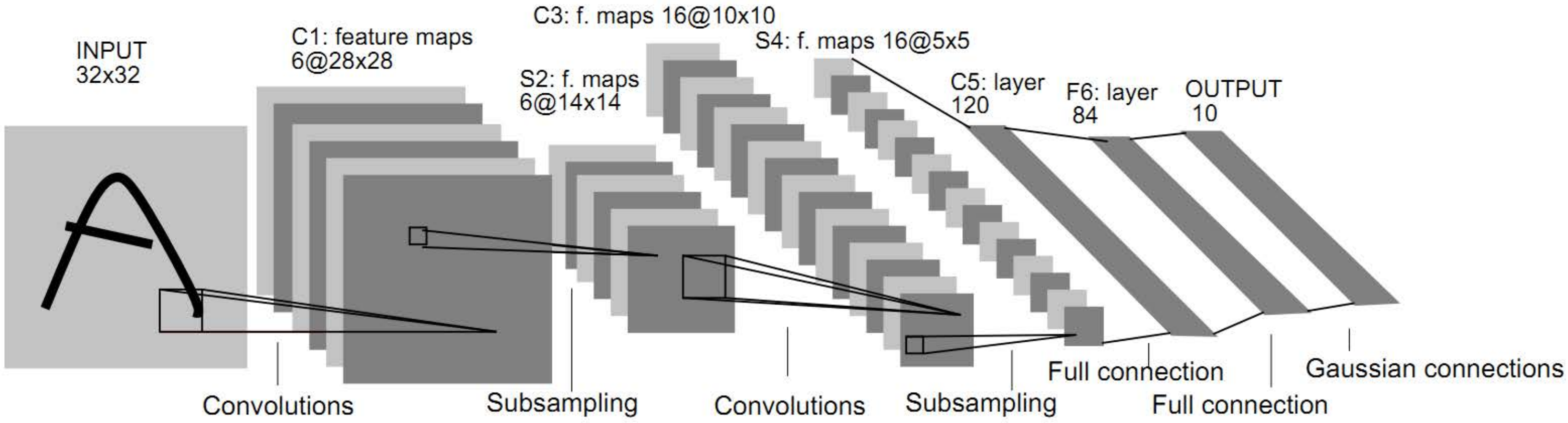


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

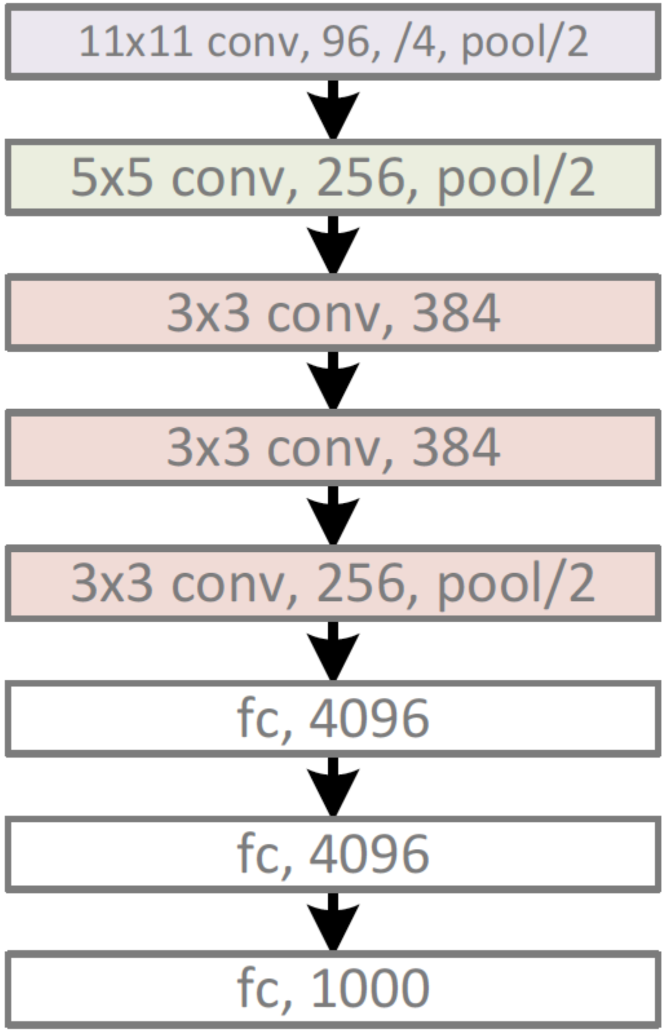
“Gradient-based learning applied to document recognition”, LeCun et al. 1998

AlexNet – 2012

8 layers: 5 conv and max-pooling + 3 fully-connected

LeNet-style backbone, plus:

- ReLU
  - Accelerate training
  - better gradprop (vs. tanh)
- Dropout
  - Reduce overfitting
- Data augmentation
  - Image transformation
  - Reduce overfitting



“ImageNet Classification with Deep Convolutional Neural Networks”, Krizhevsky, Sutskever, Hinton. NIPS 2012

VGG16/19 - 2014

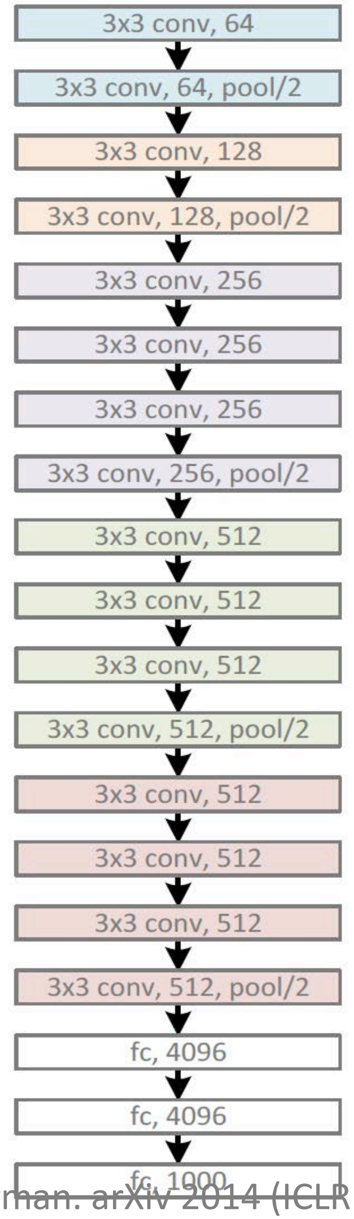
Very deep ConvNet

Modularized design

- 3x3 Conv as the module
- Stack the same module
- Same computation for each module

Stage-wise training

- VGG-11 => VGG-13 => VGG-16



3x3 conv, 64

3x3 conv, 64, pool/2

3x3 conv, 128

3x3 conv, 128, pool/2

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256, pool/2

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512, pool/2

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512, pool/2

fc, 4096

fc, 4096

fc, 1000

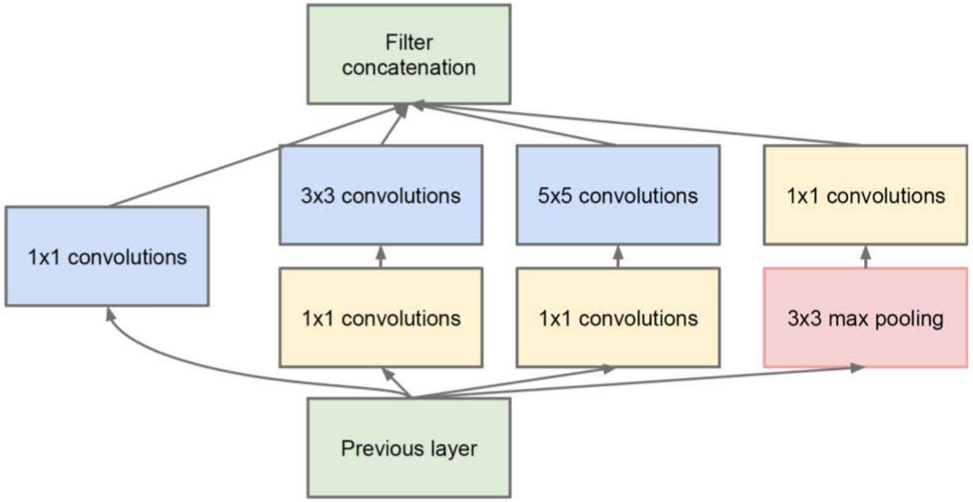
“Very Deep Convolutional Networks for Large-Scale Image Recognition”, Simonyan & Zisserman. arXiv 2014 (ICLR 2015)

GoogleNet/Inception - 2014

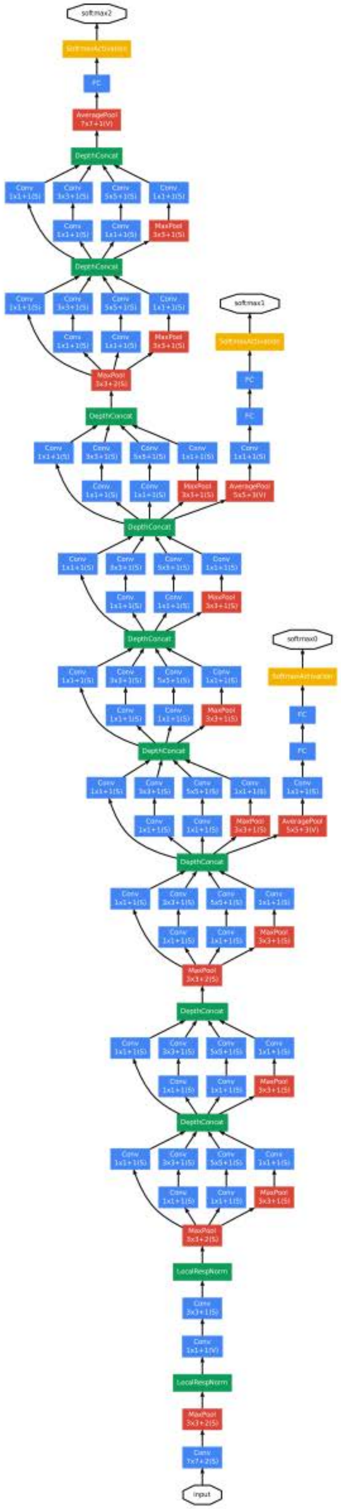
22 layers

Multiple branches

- e.g., 1x1, 3x3, 5x5, pooling
- merged by concatenation
- Reduce dimensionality by 1x1 before expensive 3x3/5x5 conv

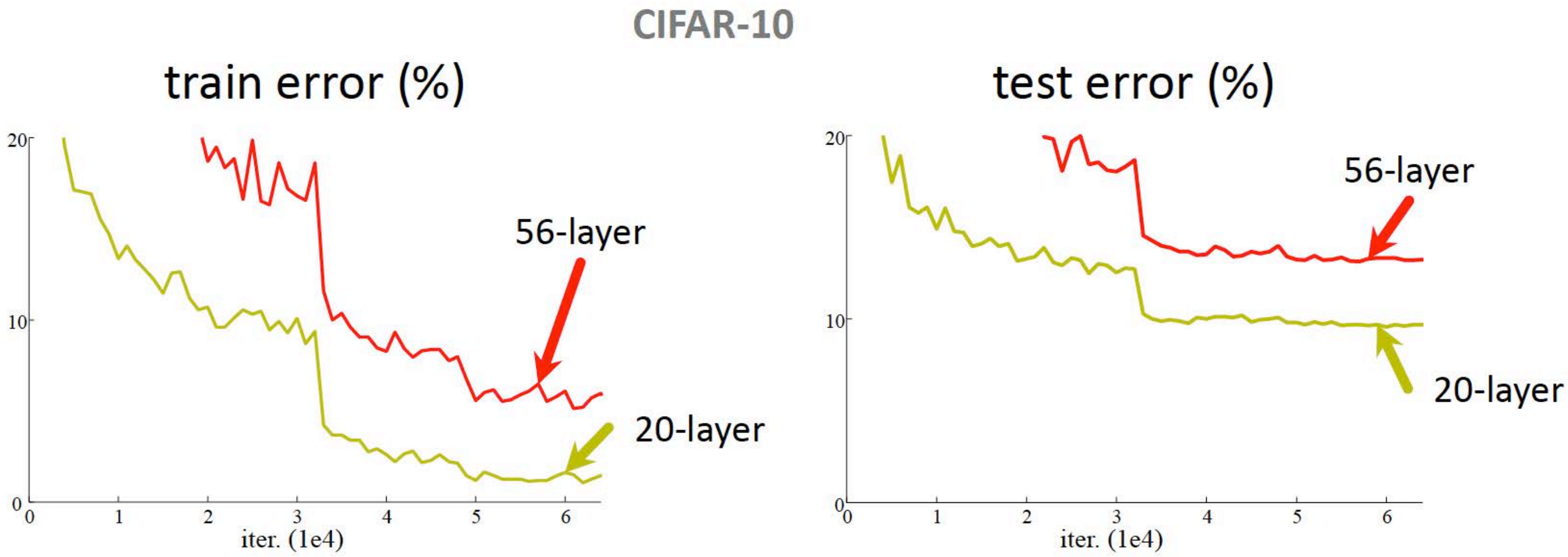


Szegedy et al. “Going deeper with convolutions”. arXiv 2014 (CVPR 2015)



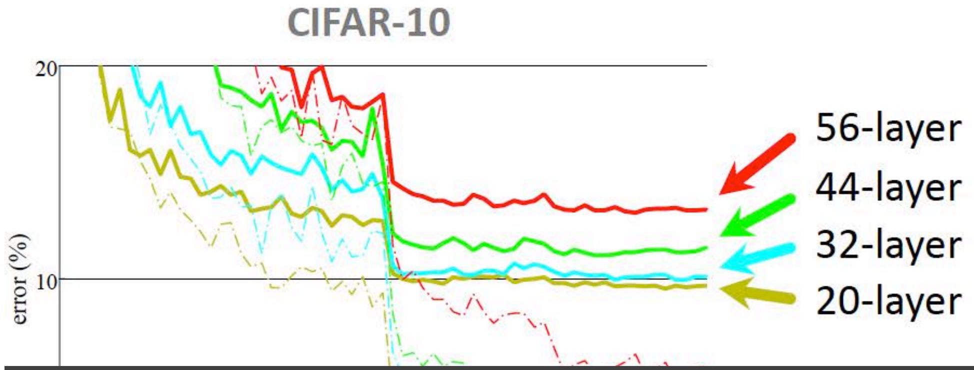
Going Deeper

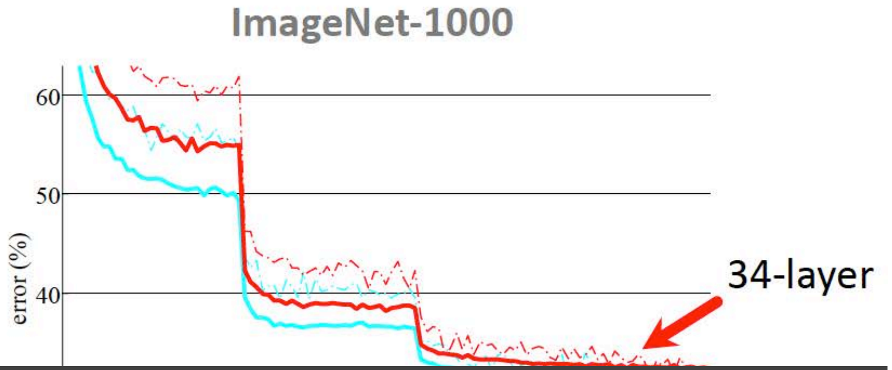
Simply stacking layers?



- Plain nets: stacking 3x3 conv layers
- 56-layer net has **higher training error and test error** than 20-layer net
- A deeper model should not have higher training error

Going Deeper





Cannot go deeper for deep neural networks!

Problem:
deeper plain nets have higher training error on various datasets

Optimization difficulties:

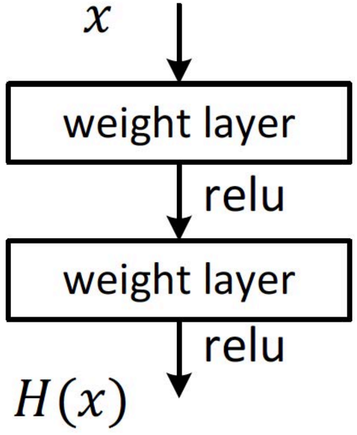
- vanishing gradient
- solvers struggle to find the solution when going deeper

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

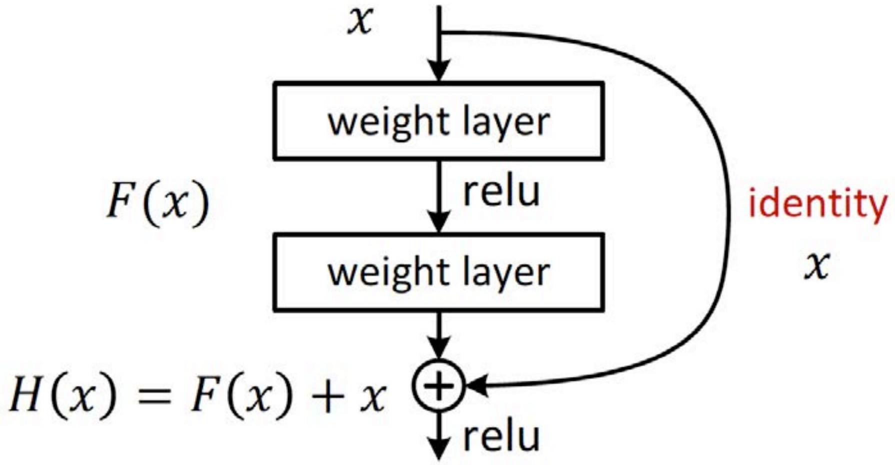
ResNets-2016

Plain net

any small subnet

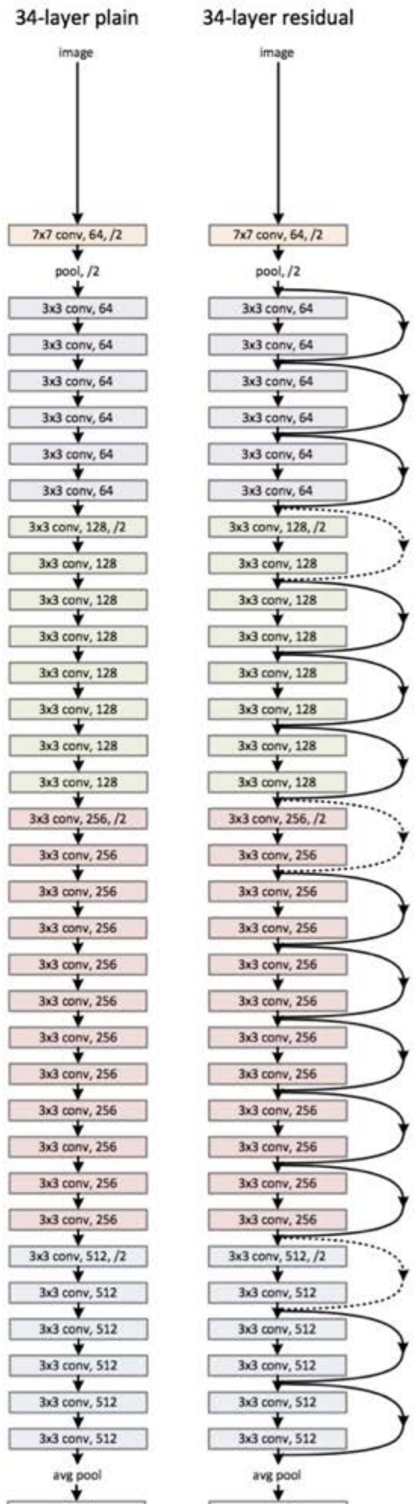


Residual net

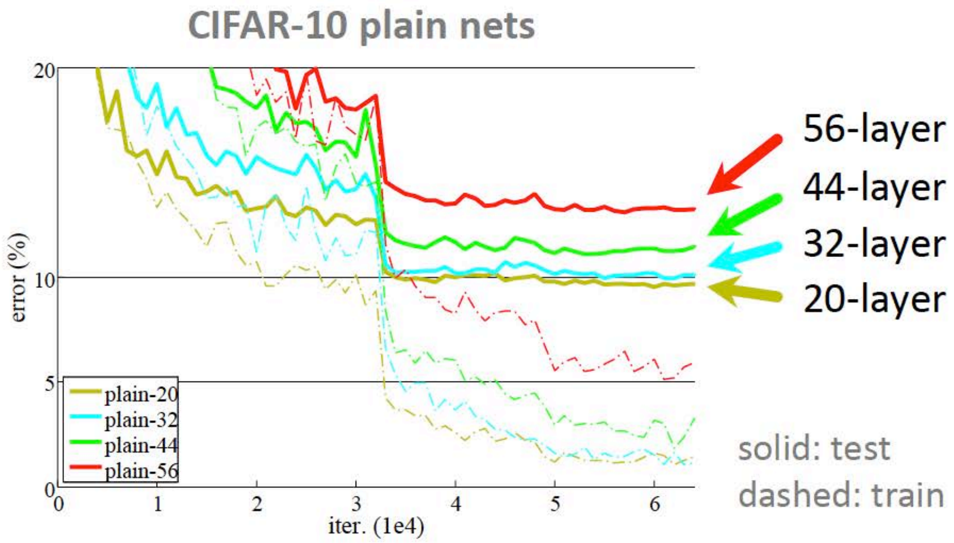


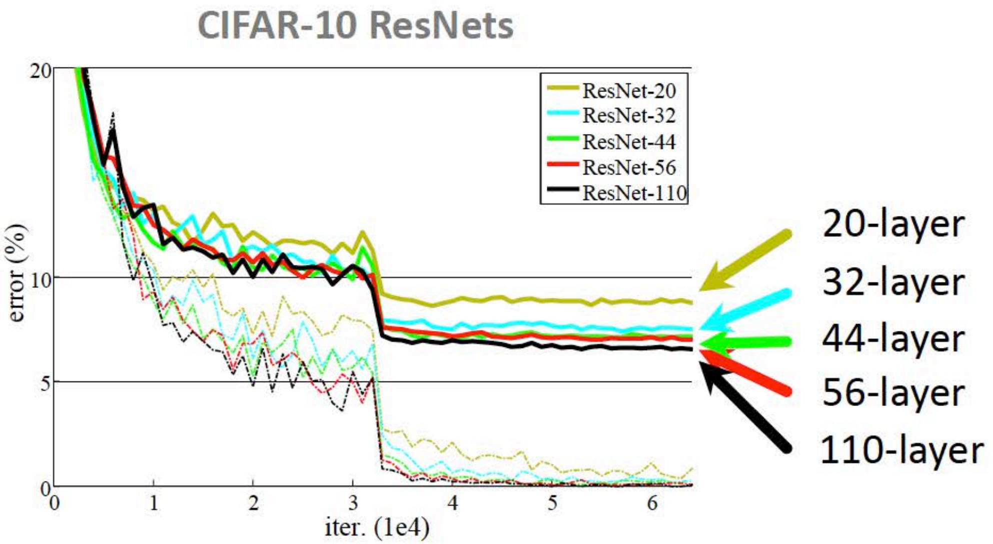
gradients can flow directly through the skip connections backwards

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Rec



ResNets-2016

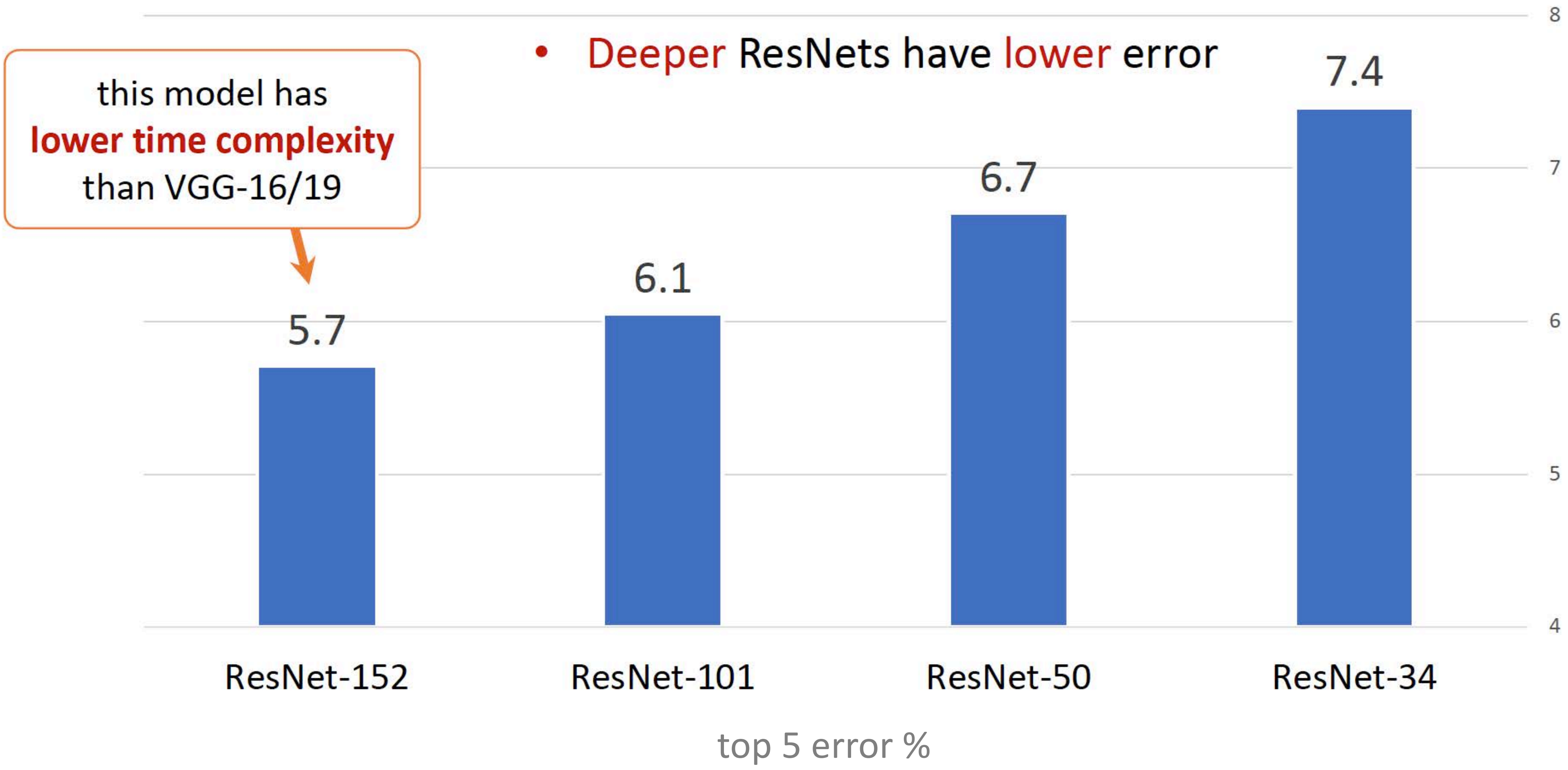




- Deep ResNets can be trained easier
- Deeper ResNets have lower training error, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

ImageNet experiments

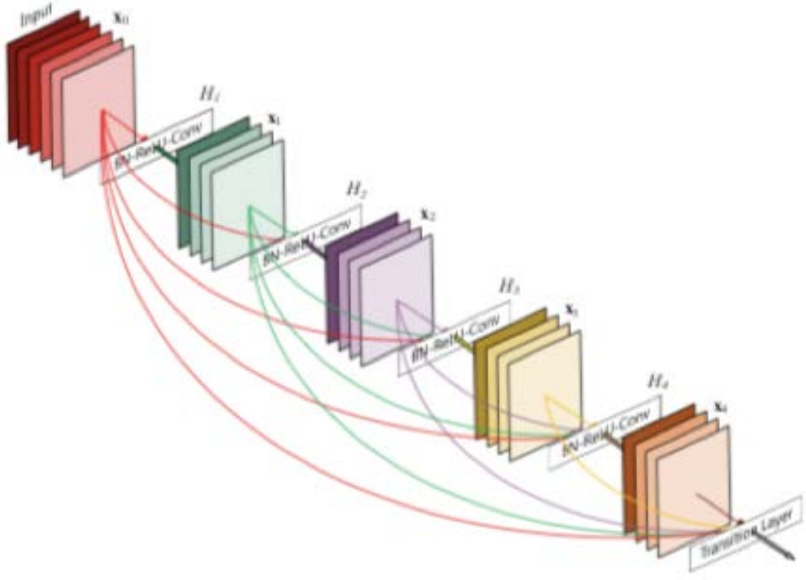


- Deeper ResNets have lower error

this model has
lower time complexity
than VGG-16/19

DenseNets - 2018

- simply connect every layer directly with each other
  - each layer has direct access to the gradients from the loss function and the original input image
  - exploit the potential of the network through feature reuse
- DenseNets concatenate the output feature maps of the layer with the incoming feature maps.



$$x_l = H_l(x_{l-1})$$

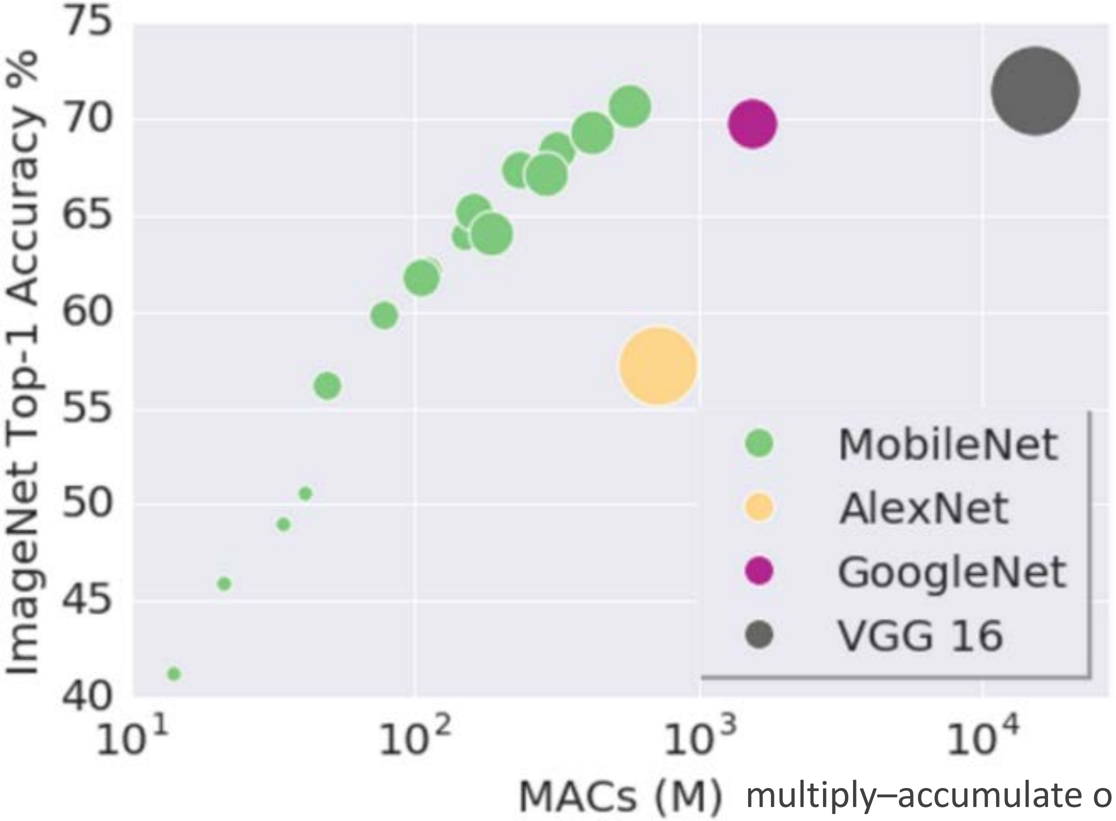
$$x_l = H_l(x_{l-1}) + x_{l-1}$$

$$x_l = H_l([x_0, x_1, \dots, x_{l-1}])$$

G. Huang, Z. Liu and L. van der Maaten, “Densely Connected Convolutional Networks,” 2018.

MobileNets - 2017

Light-weight ConvNets for mobile applications using depth-wise convolutions

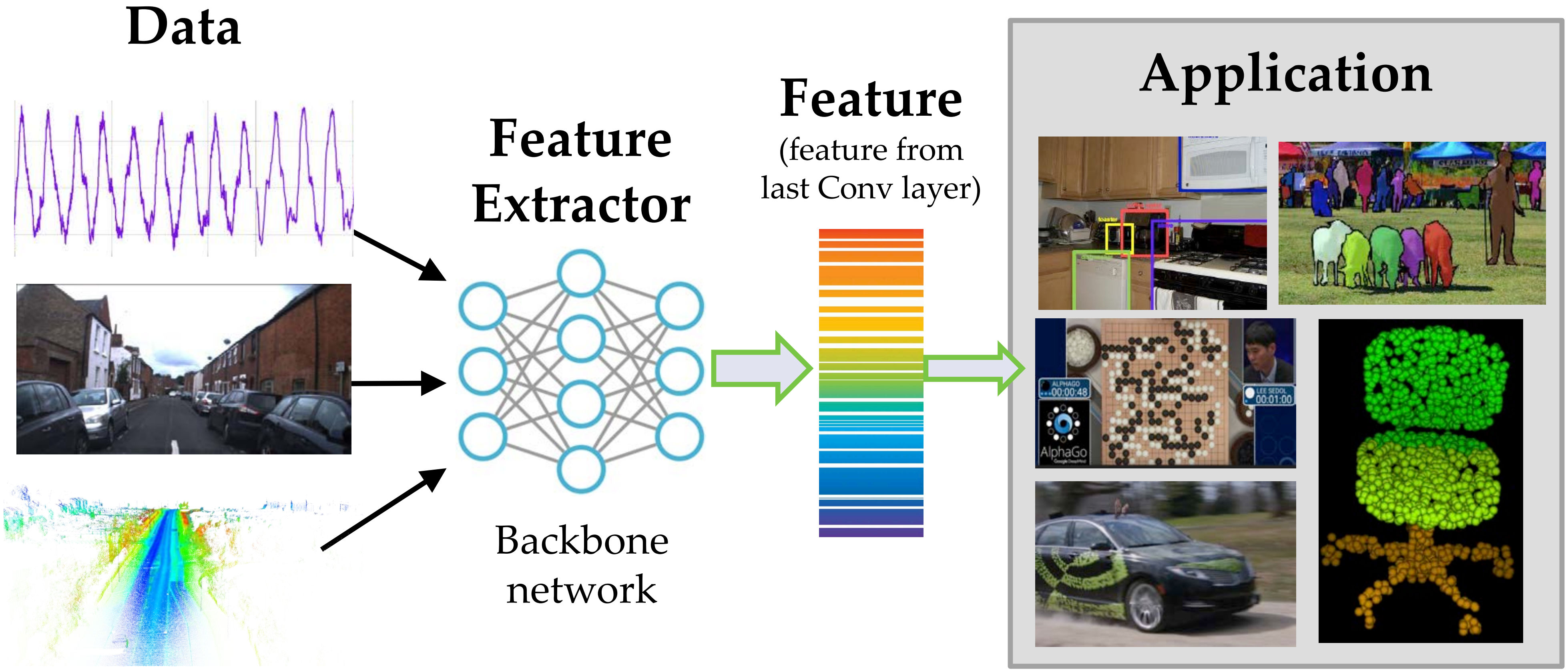


multiply–accumulate operation

Howard. et. al. MobileNets: Efficient Convolutional Neural Networks for Mobile VisionApplications 2017

Deep Learning Applications

Deep Learning Applications



Data

Feature Extractor

Backbone network

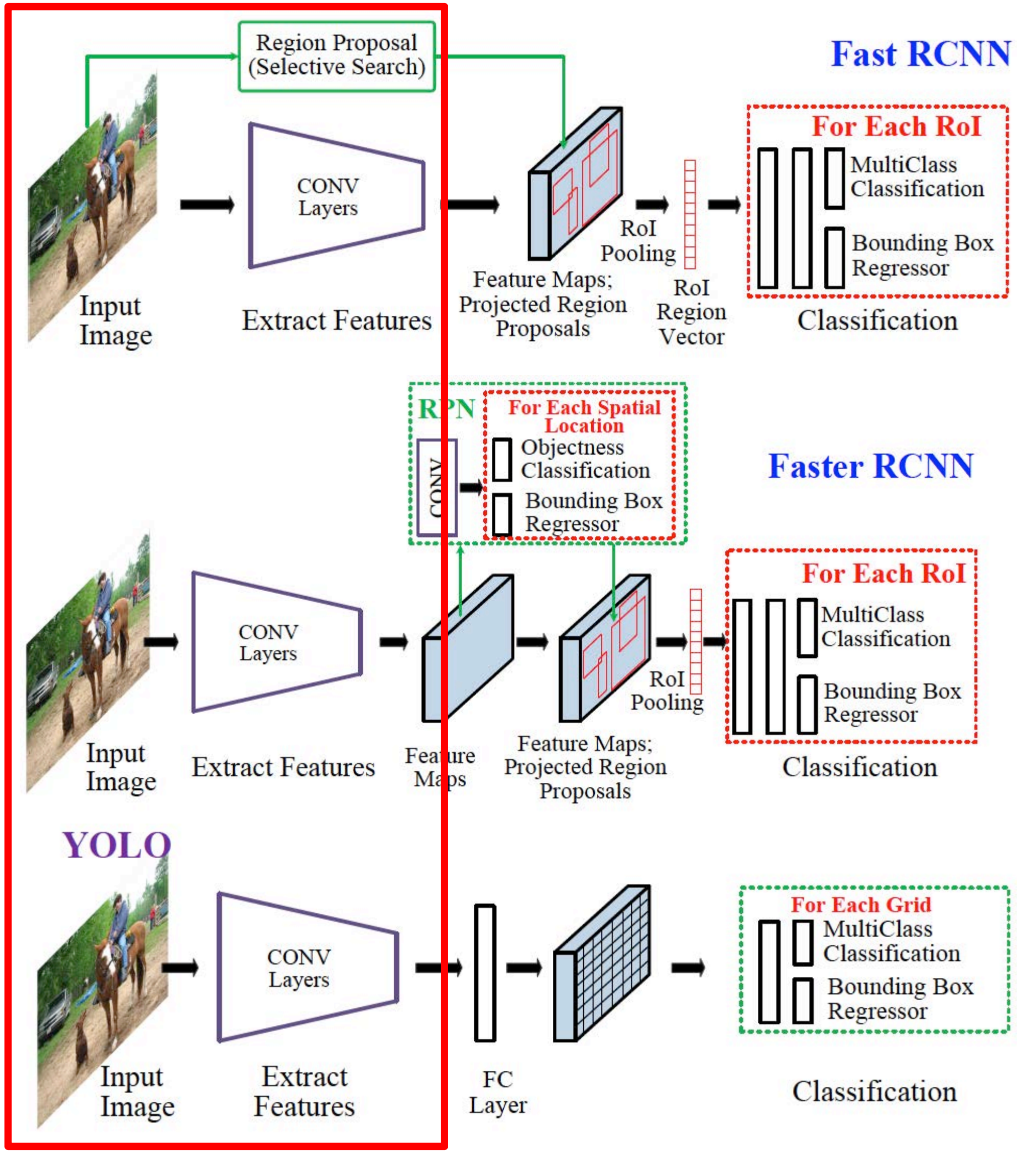
Feature

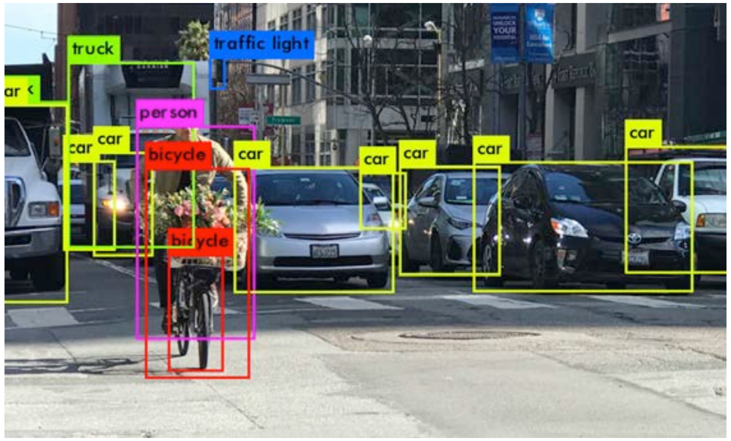
(feature from last Conv layer)

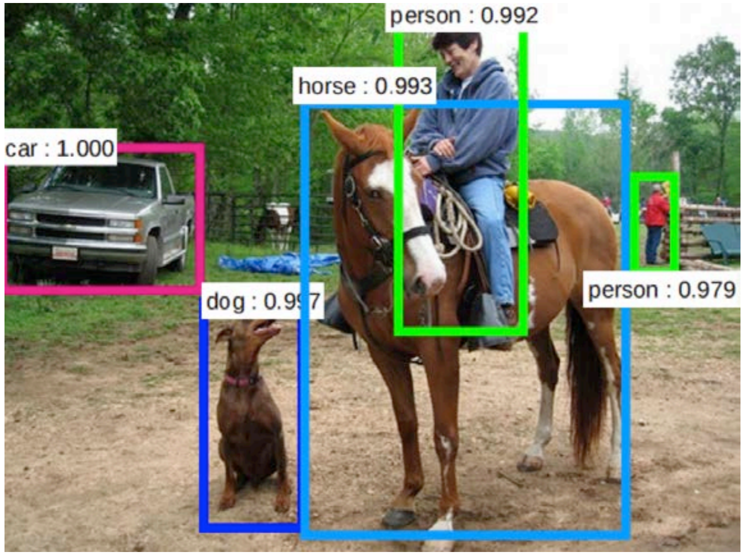
Application

Object Detection and Recognition

Vision based: RCNN, Fast RCNN, Faster RCNN, YOLO, SSD,…..

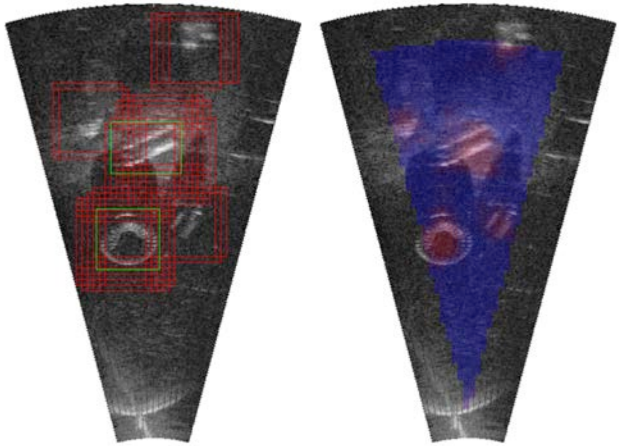




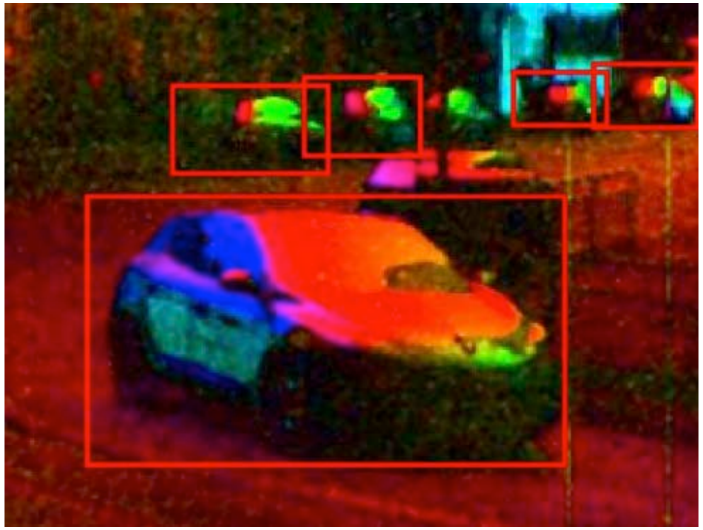


Object Detection and Recognition

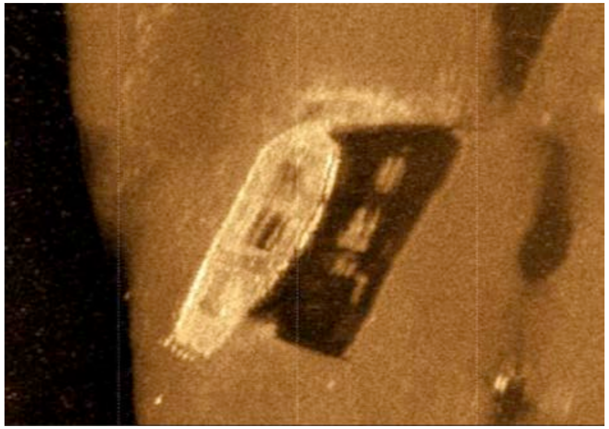
Radar and sonar based object detection and recognition



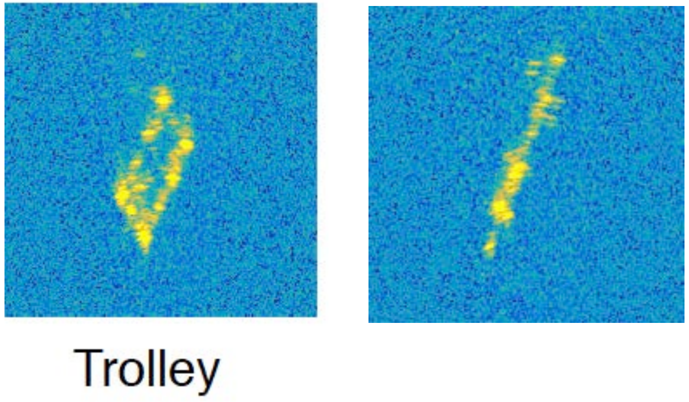
object detection from sonar images
[Valdenegro 2016]



vehicle detection using polarised infrared sensors
[Sheeny 2018]



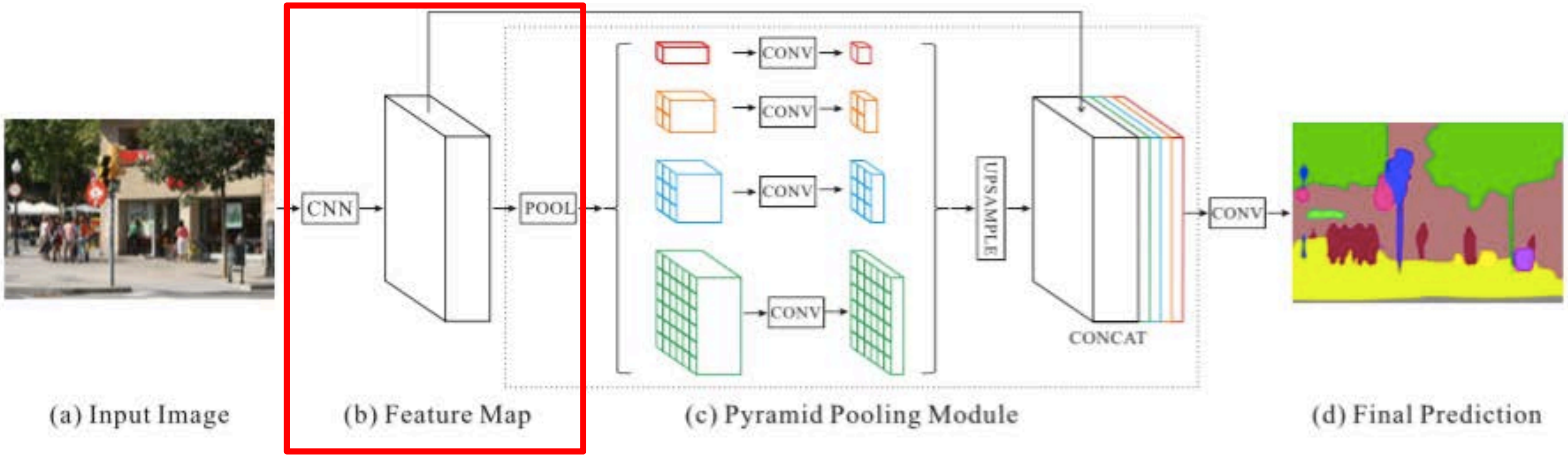
object detection/recognition on side scan



object detection/recognition on radar

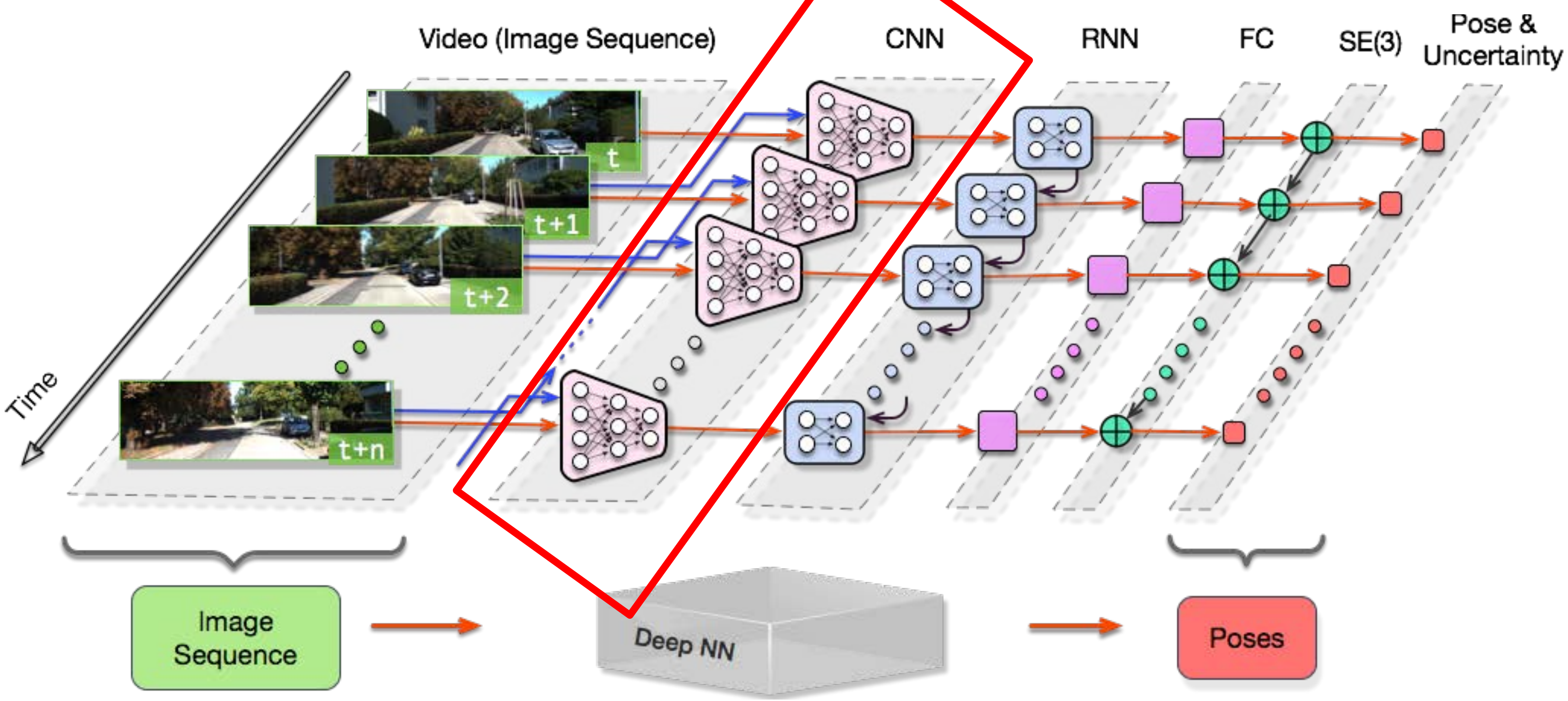
Semantic Segmentation

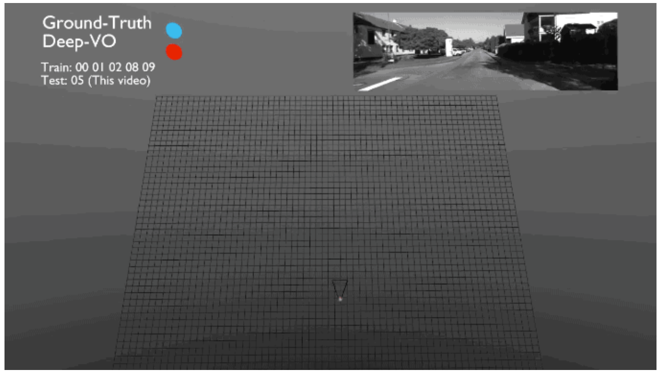
FCN, SegNet, RefineNet, PSPNet, …..



Visual Odometry

- DeepVO, UnDeepVO, VINet ……





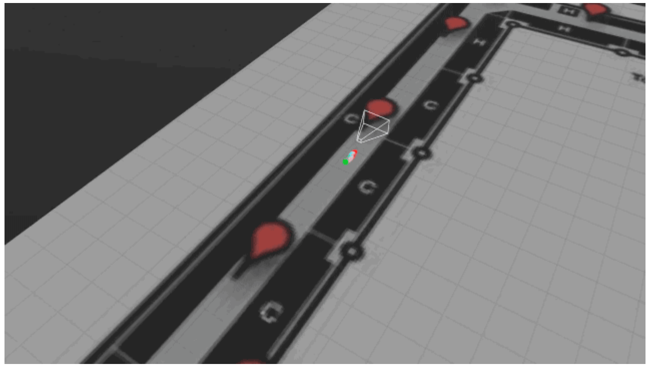
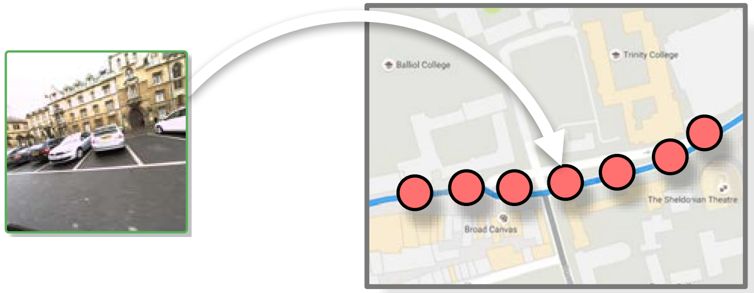
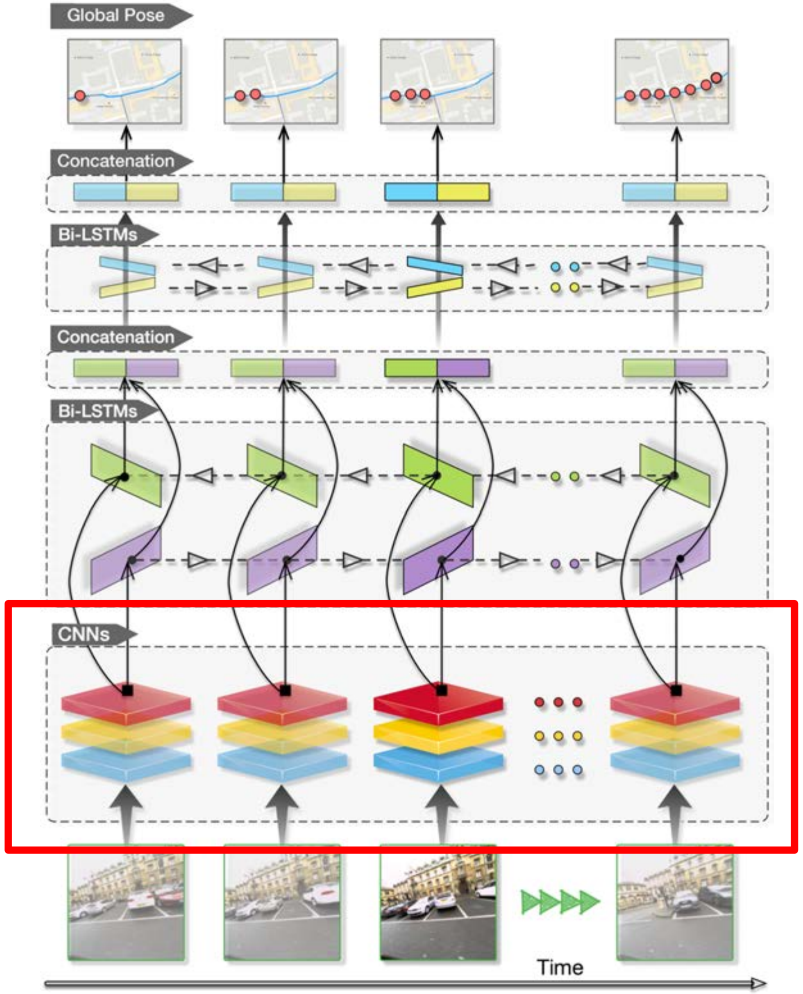
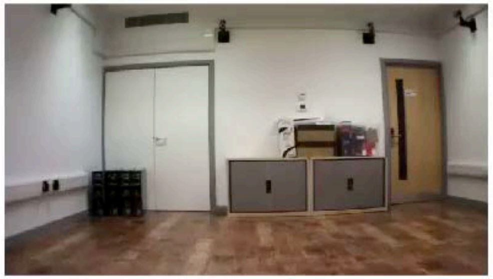


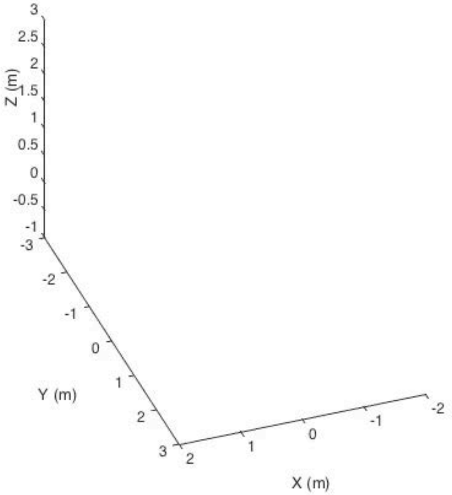
Image based Localisation

PoseNet, VidLoc, …. : map images to 6 DoF poses



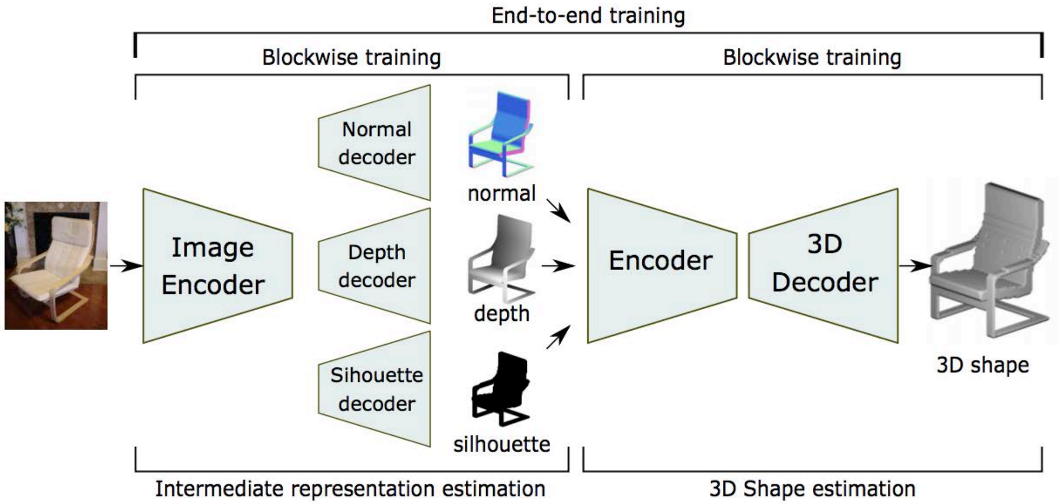






3D Reconstruction

- OctNet, Octree Generative Network (OGN), Mesh R-CNN, …


End-to-end training

Blockwise training

Blockwise training

Image Encoder

Normal decoder

Depth decoder

Sihouette decoder

normal

depth

silhouette

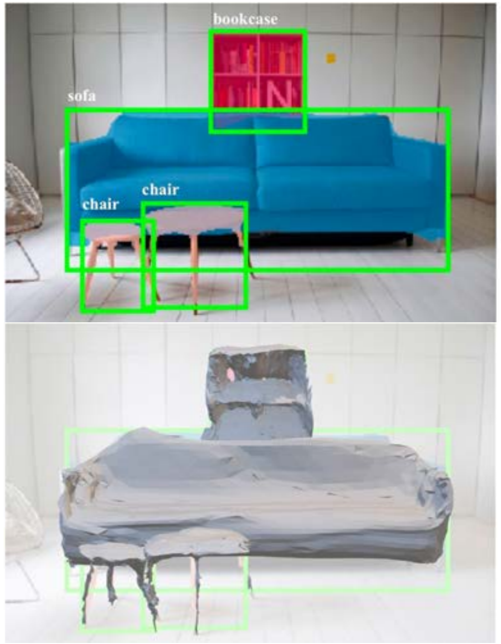
Encoder

3D Decoder

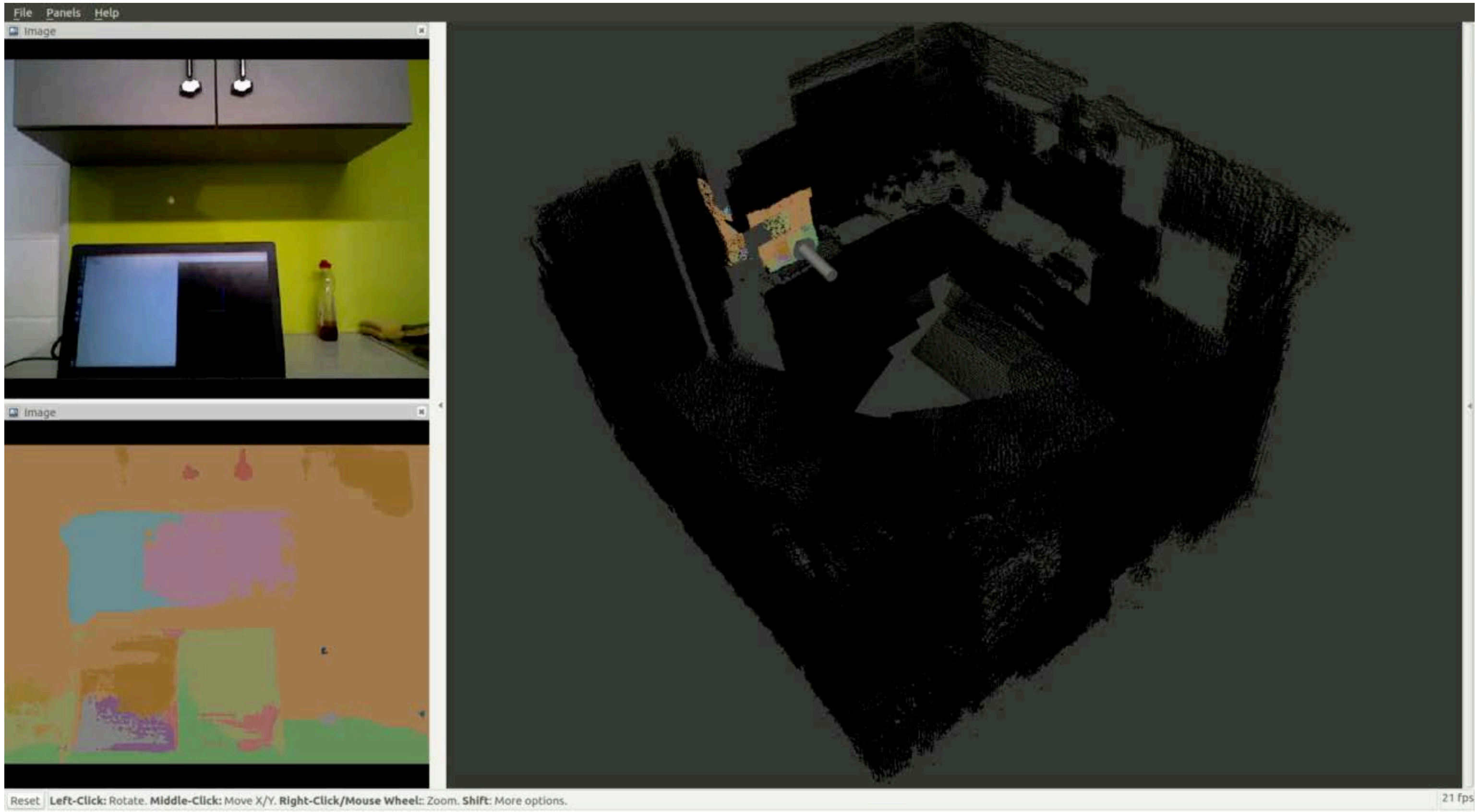
3D shape

Intermediate representation estimation

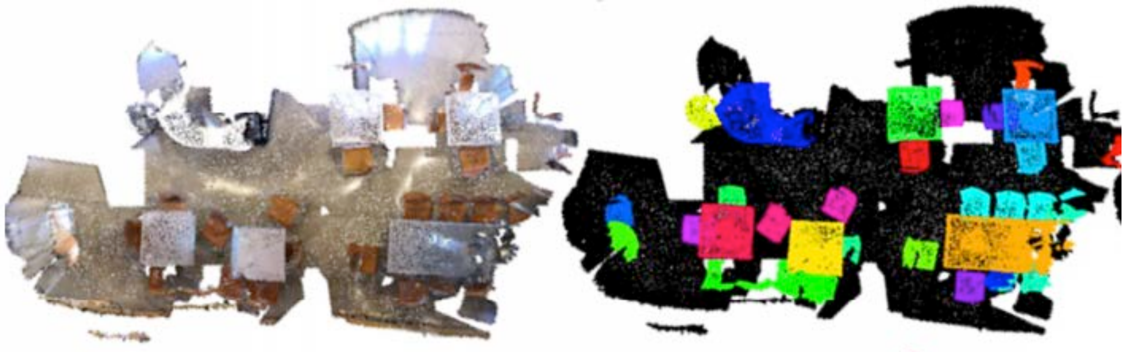
3D Shape estimation



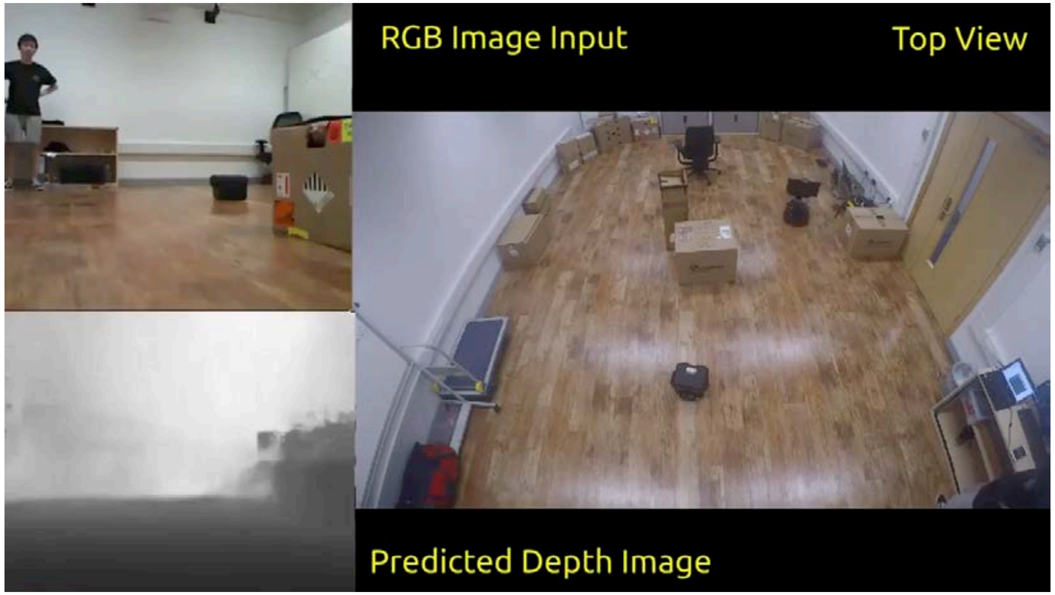
Semantic Mapping

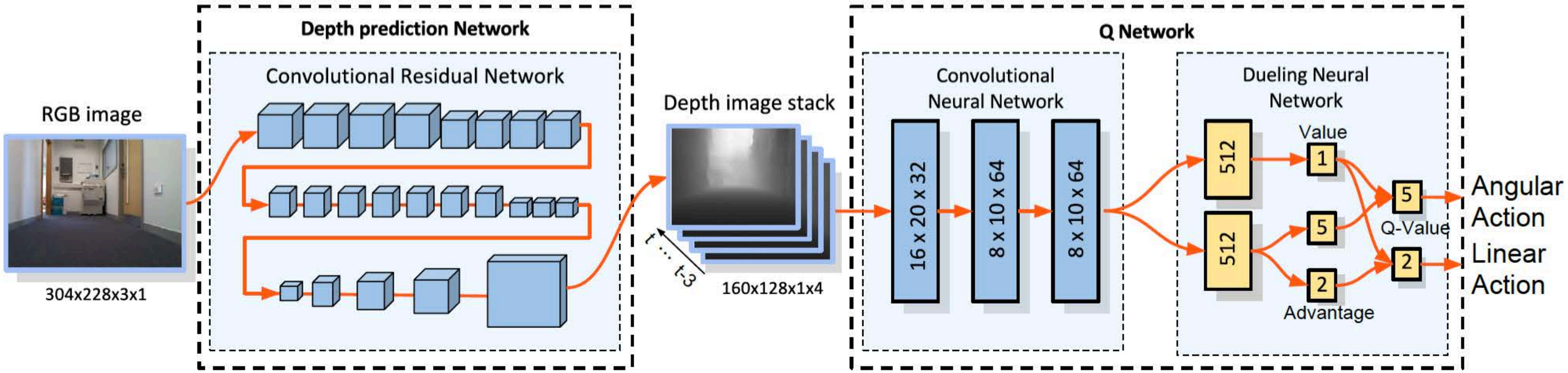


Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift:** More options.



Robot Navigation





Summary

- Deep Learning is a powerful tool
- Learning representation is the key for Deep Learning

Thank you for your attention!

Slides adapted from Andrej Karpathy, Kaiming He